高性能画像処理ＬＳＩに関する研究

昭和61年3月

福　島　　忠

高性能画像処理ＬＳＩに関する研究

昭和61年3月

福　島　　忠

目次

まえがき

画像処理・パターン認識の基礎的な研究は、１９６０年前後に大型計算機を用いて始められた。１９７０年代になると、大型計算機より性能価格比の良いミニコンピュータが出現し、これを活用して急速に画像処理の実用化が進められた。しかし、システム全体としては高価格であったため、リモートセンシングや医療機器など、一部の応用に限られていた。

そこで、画像処理の適用を広めるために、マイクロコンピュータを用いたシステムが開発されるようになるが、絶対的な処理性能が低いため、汎用的な画像処理システムの構築は非常に困難であった。このため、高性能な画像処理専用ＬＳＩ（Large Scale Integration）の開発が強く要望されるようになる。

本論文は、濃淡画像を扱える高性能な画像処理専用ＬＳＩの設計・開発上の問題点を検討し、その解決策としてのＬＳＩ、ＩＳＰ（Image Signal Processor）のアーキテクチャと、ＩＳＰを用いたシステムの構築法を示したものである。

第１章では、画像処理の手順とＬＳＩ化の目的を述べ、完全並列型と局所並列型の二つに分類して、画像処理ＬＳＩの動向およびその特徴を論じ、本研究の基本方針について述べる。

第２章では、局所並列型の画像処理ＬＳＩを開発する際の問題点を検討し、高速化と小型化の二つの観点から、カーネル拡張方式・画像走査方式・ＬＳＩの内部構成について論じ、ＩＳＰの基本アーキテクチャを明らかにする。

第３章では、第２章で明らかにした基本アーキテクチャの上で、２値・濃淡・色彩画像に対する種々の演算を実現するため、構造的な観点と機能的な観点とからアーキテクチャを検討し、演算データの制御と演算回路の構成および機能についての検討結果を述べる。

第４章では、基本アーキテクチャ上で、複数個のマスクデータを用いるマルチマスクオペレーションを実行する場合の問題点を論じ、データ供給と比較演算に対するアーキテクチャの検討結果について述べる。

第５章では、高性能画像処理ＬＳＩ、ＩＳＰを用いた画像処理システムの構築方法、およびそのシステムの性能の評価結果を論じ、さらにＩＳＰを活用して構築した実際のシステムについて述べる。

第６章では、本論文の結論を要約して示す。

第１章　序論

１．１　まえがき

電子計算機の発展にともなって、ディジタル画像処理が広い分野に適用されるようになってきた。それは、光学系のアナログ画像処理に対して、電子計算機を用いたディジタル画像処理が、処理の柔軟性・演算精度・制御の容易さ・再現性などに優れていることに加えて、演算道具としての電子計算機が安価になってきたことが、大きな要因である。このことは、メインフレームに対して、性能価格比の数段優れたミニコンピュータの出現が、画像処理アルゴリズムの研究や、画像処理技術の実用化を促したことに、顕著に現れている。最近のマイクロコンピュータの普及拡大は、この傾向を一層強めているといえる。

画像処理技術の実用化は、ミニコンピュータを用いて、リモートセンシングと医療エレクトロニクスの分野で始められた。これは、ミニコンピュータによる性能価格比の向上に負うところが大きいが、それでもなおコンピュータ本体が高価であったため、当初その応用分野は、ほとんど上記の二つの分野に限られていた。この二つの分野の次に、画像処理技術の応用分野としてクローズアップしてくるのが、装置産業として名高い半導体産業であったことも、画像処理技術が高価であったことを示している。それ故に、マイクロコンピュータの普及は、画像処理技術の産業応用に大きな衝撃を与えつつあると言える。

マイクロコンピュータを活用すると、ミニコンピュータを用いた画像処理システムより、さらに数倍、性能価格比の優れたシステムを構築できるようになった。このため、画像処理技術の応用分野も、一般産業へ急速に広まりつつある。それらは、鉄鋼、繊維、印刷、自動車など、多種の製造業に及んでいる。しかし、これらの分野への応用技術は、ディジタル画像の最小構成要素である画素(pixel)のおのおのに、１ビットの情報を持たせた２値画像の処理が主である。これは、マイクロコンピュータの低い処理能力に、その主な要因がある。

２値画像は、１画素１ビットのため、２５６×２５６画素から成る画像でも、６４キロビットと情報量は少なく、処理も簡単で取り扱いが容易である。このため、マイクロコンピュータを用いて、高速に処理できる画像処理システムを、小型にかつ安価に構築できる。しかし、テレビカメラなどの入力機器から採取される映像は、もともと中間調を有する１画素多ビットの濃淡画像であるため、１画素１ビットに圧縮してから、後の処理を行なうことになる。そのため応用分野は、計測や形状認識といった比較的処理の簡単な分野に限られている。

一方、濃淡画像は、１画素に複数ビット(通常、モノクロ画像で６～８ビット)の情報を持つため、種々の演算によりさまざまな処理が可能である。その反面、２値画像に比べ扱う情報量が多いため、必要なハードウェアや処理時間は多大になる。このため、マイクロコンピュータを用いて濃淡画像処理を実行する場合、ごく一部の画像演算を除いて、実際の応用に適する処理速度を達成することは困難である。

そこで、画像処理専用のプロセッサを開発して、処理の高速化を図ることになる。しかし、濃淡画像の情報量が多いため、専用プロセッサのハードウェア規模そのものが大きくなり、多種の濃淡画像処理機能を、小型でかつ安価に実現することは困難である。このた

め、ほとんどの専用プロセッサは、特定分野向けのものとなっている。このことは、画像処理技術を新たな分野へ応用しようとする時、新しい専用プロセッサを開発しなければならず、画像処理技術の普及拡大の大きな障害となっている。

つまり、今後、画像処理技術を一般産業へより広く普及させるためには、２値画像のみならず、より高度な処理が可能な濃淡画像の種々の機能を実行し得る「汎用プロセッサ」を開発する必要がある。そのためには、濃淡画像を高速に処理できる「画像処理用ＬＳＩ」が不可欠である。なぜなら、高性能な画像処理用ＬＳＩとマイクロコンピュータを併用することにより、初めて種々の画像演算を高速に実行できる汎用画像処理システムを、小型でかつ安価に構築できるからである。

本章では、上記の観点から、画像処理用ＬＳＩの開発目的と、その動向と特徴について考察し、本研究の基本方針について論ずる。

１．２　画像処理の流れとＬＳＩ化の目的

画像処理技術は、要素技術として高い汎用性を備えている。処理の対象となるものは、写真・文書・図形・風景など、ありとあらゆる２次元情報におよぶ。このため、技術コストの低下に伴って、さまざまな分野へ応用されつつある。

一般に、２次元のディジタル情報を扱うディジタル画像処理は、二つの大きな技術内容を包含している。一つは、画質の劣化を復元したり、幾何学的な歪みを補正したりする画像変換（狭義の画像処理と呼ばれることがある）であり、もう一つは、画像の構造を解析したり、特徴を抽出したりする画像認識（image recognition）・画像理解（image understanding）である。この二つを包含する画像処理の基本的な流れは、図１．１のように示すことができる。

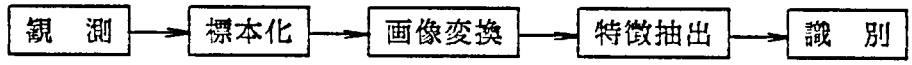


図１．１　画像処理の基本的な流れ

もちろん、画像に対する処理の目的は千差万別であり、図１．１の流れに沿ってすべての画像が処理できるわけではない。実際には、フィードバックを含む認識アプローチ［１］なども提案されている。しかし、画像処理技術を一般産業へ応用する観点から考えると、まず図１．１の処理の流れに沿った認識手法を考察するのが最善と思われる。なぜなら、これまで研究されてきた多くのアルゴリズムを活用できる上、汎用的な手法を得やすいからである。

図１．１の画像処理の流れは、製品の外観検査を例にとると、次のように表現できる。まず観測ステージで、ラインスキャナやテレビカメラなどを用いて、対象物を含む２次元情報を採取する。採取された情報は通常アナログ量であるため、標本化ステージで、アナログ／ディジタル変換回路（Ａ／Ｄ converter）を用いてディジタル化される。ディジタル化された情報は、雑音除去・エッジ強調・２値化・細線化などといった画像変換処理（前処理とも呼ばれる）の後、幾何学的特徴計測・色彩パラメータ計測・領域分割といった特徴抽出処理を受ける。そして識別ステージで、抽出された特徴情報から製品の良否を判定することになる（特徴抽出および識別ステージは、一般にパターン認識と呼ばれている）。つまり、「観測」された膨大な２次元情報は、最終的には良否というわずか１ビットの情報に圧縮されることになる。より現実的には、留保・追加検査といった場合を含めて、せいぜい２～３ビットの情報におとされることになる。このことをうまく表現したものに図１．２がある［２］。

図１．２における「低レベル処理」(low-level processing) とは、特徴や記号表現を得る以前の画像の処理であり、図１．１における画像変換と特徴抽出処理に相当する。また、「高レベル処理」(high-level processing) とは、識別処理に他ならない。図１．２からも分かるように、いかなる高レベル処理の枠組を用いる場合にも、低レベル処理は不

可欠であり、重要な情報だけを抽出して冗長な情報を棄てる必要がある。低レベル処理の性能が、画像認識システム全体の性能を左右するといっても過言ではない。しかるに、膨大な量のデータをいかに高速処理するかということが重要な問題となる。その一つの解が並列処理である。

並列処理においては、同一の演算を並行して処理する方式ＳＩＭＤ(Single Instruction Multiple Data stream) と、異なった演算を並行して実行する方式ＭＩＭＤ (Multiple Instruction Multiple Data stream) があるが、画像処理は前者が一般的である。図１．１において、観測された画像情報は標本化ステージで、画素（pixel）と呼ばれる最小単位の情報の集合体に変換される。この段階では個々の画素情報は、それ単体としてよりもむしろ周囲の画素情報との間に、より多くの情報を持っている。そのため、低レベル処理においては、すべての画素データに一様な演算を施すことが多い。これが、ＳＩＭＤ型の並列処理に適しているのである。

大規模な並列処理の基本概念は、Unger［３］によって提案されているが、膨大なハードウェアを必要とするものである。多大のハードウェアは、システムコストを増加させるだけでなく、システムの信頼性を著しく低下させる。低レベル処理のＬＳＩ化は、この二つの欠点を解決するものである。換言すると、ＬＳＩ化により、大規模な並列処理を実行できる画像処理システムを、小型で信頼性よく構築することが可能になるのである。

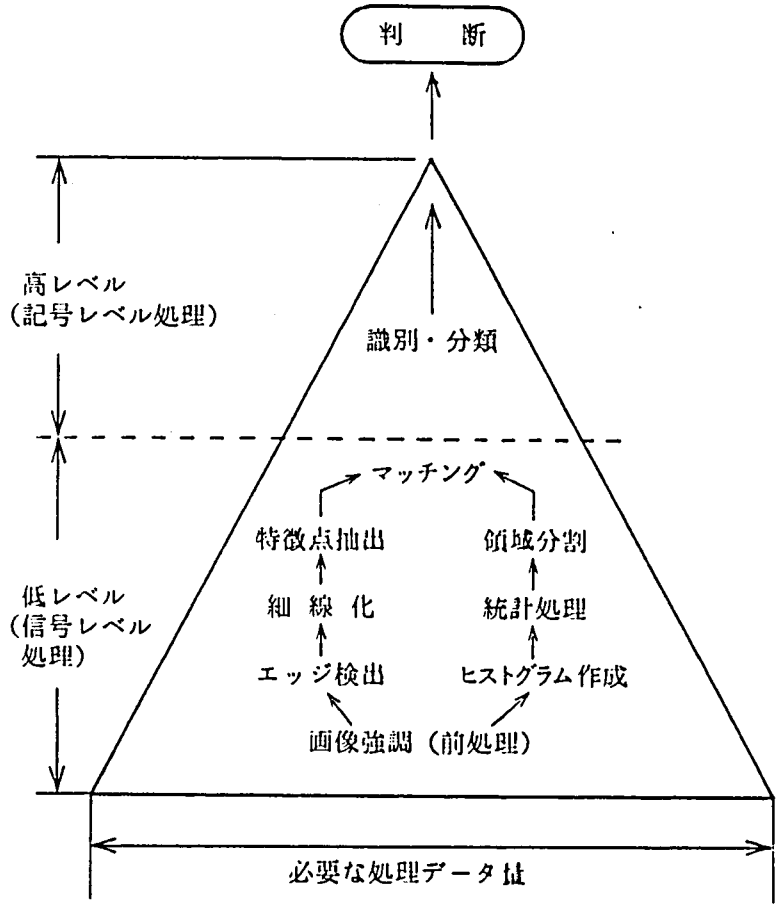


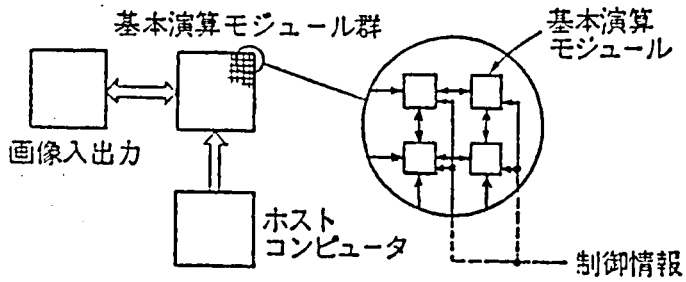
図１．２　画像認識・理解の階層性

１．３　画像処理ＬＳＩの動向と特徴

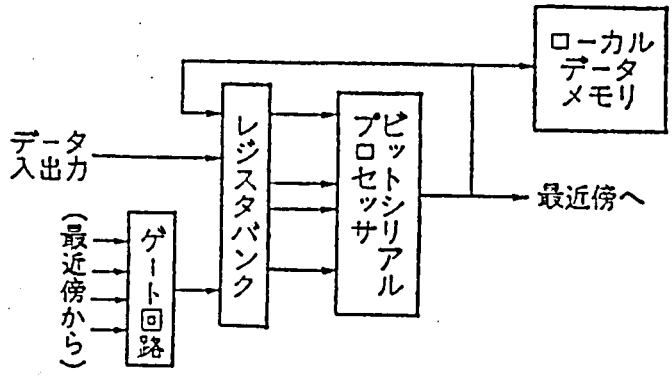
画像を高速に処理する画像処理専用ＬＳＩの出現への期待は、半導体技術の進歩に伴い、ＬＳＩの集積度が飛躍的に向上し始めた１９７０年代後半頃から急速に高まった。最初の画像処理専用ＬＳＩが、１９８０年に英国ロンドン大学で開発されて、その後いくつかの試みがなされている。その流れは、大きく二つに分類される。一つは完全並列型（Ｆully Ｐarallel Ｐrocessor）と呼ばれるものであり、もう一つは局所並列型（Ｐartially Ｐarallel Ｐrocessor）と呼ばれるものである。本節では、上記の二つのタイプについて、それぞれのアーキテクチャ上の特徴、および問題点について考察する。

１．３．１　完全並列型

完全並列型と呼ばれるタイプは、全画像のそれぞれの画素に、基本演算モジュールであるＰＥ（Ｐrocessor Ｅlement）を一つずつ用意して、全画素を並列に演算するものである。図１．３に示すように、ＰＥは画素と同じ２次元構造に配列されている［４］。この方式の基本原理は、古くはUngerによって提案され［３］、これまでに、ＣＬＩＰ（Cellular Logic Ｉmage Ｐrocessor)－４［５］、ＤＡＰ（Distributed Array Ｐrocessor）［６］、ＭＰＰ（Massively Ｐarallel Ｐrocessor）［７］、ＡＰＰ（Ａdaptive Ａrray Ｐrocessor）［８］などが開発されている。いずれも、基本的には図１．３のような共通のアーキテクチャを持つ。このタイプのＰＥは、ビットシリアル型論理演算プロセッサと、ビット単位のアクセスが可能なローカルデータメモリから構成されており、データメモリの深さに相当する枚数の２値画像を蓄えることができる。濃淡画像はビットプレーンとして扱う。表１．１に、これら完全並列型プロセッサの仕様を示す。



（ａ）典型的なアーキテクチャ



（ｂ）基本演算モジュールの構成

図１．３　完全並列型プロセッサの概念

表１．１　代表的な完全並列型プロセッサ

プロセッサ	ＰＥ数	ＰＥ間の接続	ローカルデータメモリ量	命令サイクル	ＰＥ数／チップ	使用チップ数
ＣＬＩＰ－４	９６×９６	８近傍	３２ビット	１０μｓ	８	１４４
ＤＡＰ	６４×６４	４近傍	４Ｋビット＊	２００ｎｓ	１６	２５６
ＭＰＰ	１２８×１２８	４近傍	１Ｋビット	１００ｎｓ	８	２１１２＊＊
ＡＡＰ	２５６×２５６	８近傍	９６ビット	９０ｎｓ	６４	１０２４

＊　ローカルデータメモリは別チップ。

＊＊　ＰＥに１２８×４個の冗長度がある。

完全並列型プロセッサの長所は、超高速処理とＬＳＩ開発の容易さである。このタイプのプロセッサは、画像を構成するすべての画素を同時に演算できるため、超高速処理が実現できる。命令サイクルが１０μｓと比較的処理速度の遅いＣＬＩＰ－４においても、デレビカメラから入力される画像を、実時間で細線化が実行できる。また、２次元に配列されるＰＥは、すべて同一のアーキテクチャであるため、回路の規則性が非常に高くなり、論理規模にかかわらず設計に要するマンパワーはほぼ一定で、ＬＳＩ開発の上からは理想的といえる。

一方、本タイプのプロセッサの最大の問題点は、プロセッサアレイと外界との間のデータ入出力である。画像を構成する画素のおのおのに、１個のＰＥを用意する本方式では、扱う画像のサイズに比例して用意すべきＰＥ数も増える。１チップに搭載できるＰＥ数には、集積度の点から限度があり、さらに製造コストを適当な範囲に抑える上から、システムに使用できるチップ数にも限度がある。このため、一度に処理できる画像のサイズは自ずと制限される。現在稼動しているシステムでは、ＭＰＰが最も大きな画像を完全並列処理でき、そのサイズは１２８×１２８画素である。一方、処理対象としての画像のサイズに一般的なものはなく、大きいものでは衛星画像の15,000×15,000画素サイズのようなものまである［9］。このような大きな画像を扱う場合は、全体の画像をプロセッサアレイが扱えるサイズに分割して処理することになる。プロセッサアレイと外界とのデータ入出力が、この分割処理のオーバーヘッドを左右し、システムの処理性能を決定することになる。また、画面の分割処理は、部分画像間の境界問題をも生み出すことになる。

表１．１に示した完全並列型プロセッサでは、それぞれ異なった画像データの入出力方式を採用している。ＣＬＩＰ－４では扱う画像のサイズを９６×９６に固定し、同じサイズのバッファメモリを設けることによって、ディジタル化されたテレビカメラの映像を、常にプロセッサアレイに供給し、またその処理結果を常にモニタ上に表示できる構成を実現している。　ＭＰＰでは、ステージングメモリと呼ばれる大型の画像メモリをバッファとして用いることにより、アレイサイズを超える大きな画像を扱えるようにしている。ＤＡＰは、ホストコンピュータ（ＩＣＬ２９００）の主記憶の一部を、アクティブストアとしてＤＡＰと共有するという方法を採用している［4］。

また、一般産業への応用という観点から考察すると、完全並列型プロセッサの問題点は製造コストの高さである。現在、工業応用においては、２５６×２５６サイズのノンインタレース（非飛び越し走査）画像が一般的である。このサイズの画像を完全並列処理するには、チップ当たりのＰＥ数の最も大きいＡＡＰでも1024個のＬＳＩが必要になる。工業分野における画像処理応用は、製品の組み立てや検査など省力化が主な目的のため、1000個に及ぶような多数のＩＣチップを必要とする装置はとても実用に値しない。そこで、１チップに多数のＰＥを搭載することから、集積度の一層の向上が期待されるが、１チップに搭載できるＰＥ数が増えると、ＰＥ間を接続するピン数が増加するため、高価なパッケージが必要となり、やはりＬＳＩの価格を押し上げることになってしまう。つまり一つのシステムに使用できる専用ＬＳＩの数は、チップ単価が飛躍的に低下しても、せいぜい十個程度と考えるべきである。

１．３．２　局所並列型

局所並列型と呼ばれるタイプは、平滑化、微分、細線化などの局所処理（各画素とその近傍の画素の値を使う計算処理）において、対象とする局所領域の画素データのアクセスと演算とを、それぞれ並列化によって高速化し、その局所処理を全画面に対し逐次的に走査実行するものである。局所領域の大きさは、３×３～16×16程度のものが多い。図１．４に局所並列処理の概念を示す［４］。

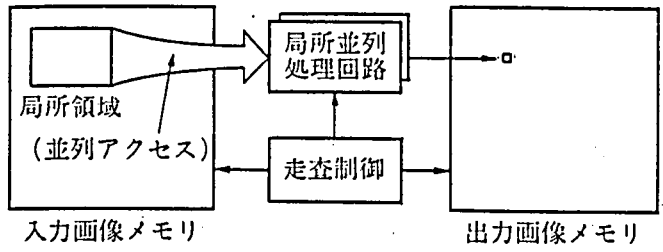


図１．４　局所並列型プロセッサの概念

このタイプに属する画像処理専用ＬＳＩとしては、米国画像理解プロジェクトにおいて、ヒューズ・エアクラフト社研究所が試作した３種類のテストチップがある。これは、ＣＣＤ（Charge Coupled Device）映像素子と同じチップ上に、やはりＣＣＤで構成したアナログ画像処理部を組み込もうとするものである。処理機能としては、微分フィルタ、コンボリューション、メディアンフィルタなどがあり、ＣＣＤもしくはビジコンカメラからの、５１２×５１２画素サイズ（３０フレーム／秒）の画像を実時間で処理できる。表１．２に、このテストチップの処理機能とその性能を示す［２］。

局所並列型プロセッサの長所は、おのおのの画像処理機能については小規模のハードウェアで実現できることと、パイプライン処理を併用して更に高速化を図れることである。

局所処理における演算領域（kernel－以下カーネルと呼ぶ）は、通常３×３もしくは５×５程度である。今、カーネルを３×３とすると、局所並列処理は図１．５のような回路により実現できる。図１．５において、入力画像データはノンインタレース走査により走査され、２本の遅延ラインを介して基本演算モジュールであるＰＥに入力される。それぞれの遅延ラインは、画像メモリの一走査線に相当する遅延を画像データに与えるため、２次元に配列された９個のＰＥにカーネルを切り出すことが可能となる。ＰＥの演算結果は統合回路（integration circuit）で統合されて出力画素データとなる。カーネルサイズを大きくするには、遅延ラインとＰＥ数を増加すればよい。また、大きな画像を処理するには、遅延ラインの遅延段数を大きくすることで対応できる。

またパイプライン処理とは、画像データを、パイプライン状に接続した処理モジュール群に送り込み、ある遅延時間後、連続して出力を得る処理方式である。画像処理においては、①オペレーションレベルのパイプライン処理と、②タスクレベルのパイプライン処理の二つの方法により高速化が図れる。オペレーションレベルのパイプライン処理とは、ある画像オペレーション（例えば、微分やＦＦＴ等）を構成する基本演算（例えば乗算、加

表１．２　米国画像理解プログラムで開発されたテストチップの処理機能と性能

Test Chip Numbers	Algorithms Implemented	Kernel Size	Operations per Pixel	Effective Operation Rate
I	Edge detection	3 x 3	16	80 KOPS
	High-pass spatial filter	3 x 3	18	90 KOPS
	Laplacian	3 x 3	13	65 KOPS
	12 dB/aperture corrector	3 x 3	18	90 KOPS
II	Sobel	3 x 3	16	32 MOPS
	Mean	3 x 3	9	18 MOPS
	Unsharp masking	3 x 3	13	26 MOPS
	Binarization	3 x 3	10	20 MOPS
	Adaptive stretch	3 x 3	12	24 MOPS
III	Laplacian	3 x 3	13	91 MOPS
	Mask programmable convolution	7 x 7	98	636 MOPS
	Programmable convolution	5 x 5	50	350 MOPS
	'Plus' shaped median	5 x 5	625	$\sim 10^3$ MOPS
	Bipolar convolution	26 x 26	1352	$\sim 10^4$ MOPS

KOPS: Kilo Operations Per Second

MOPS: Million Operations Per Second

算、画像データへのアクセス等）の、パイプライン動作により高速化するものである。一方、タスクレベルのパイプライン処理とは、画像処理オペレーションのシーケンスをパイプライン動作させ、ある画像処理タスクの実行を高速化するものである。米国画像理解プロジェクトにおける１チップイメージプロセッサは、タスクレベルのパイプライン処理の実現を目指しているといえる。

局所並列型プロセッサの短所は、処理速度と機能の柔軟性にある。

本タイプのプロセッサは、全画面に対して局所処理を逐次的に実行する必要上、画像メモリと密に結合される。このため、処理速度の上限がメモリバスの転送速度により決定されることになる。メモリバスは１６もしくは３２ビット構成で、その転送速度は、通常４０ＭＢＰＳ（Mega Bytes Per Second）程度なので、プロセッサの処理速度の上限は５００ＭＰＯＰＳ（Million Pixel Operations Per Second）のオーダーとなる[9]。つまり、これがオペレーションレベルのパイプライン処理の限界といえる。

一方、処理速度を高速化するオペレーションレベルのパイプライン処理は、ハードウェアの構成を硬直化するため、さまざまな演算を実行するという機能の柔軟性という点からは、マイナスに作用する。オペレーションレベルのパイプライン処理を押し進めると、特定サイズのカーネルに対する特定の機能しか実行できなくなるからである。この意味では、タスクレベルのパイプライン処理が有効であるが、これはタスク変更のための手続きが、無視できないほど大きなオーバーヘッドとなる。

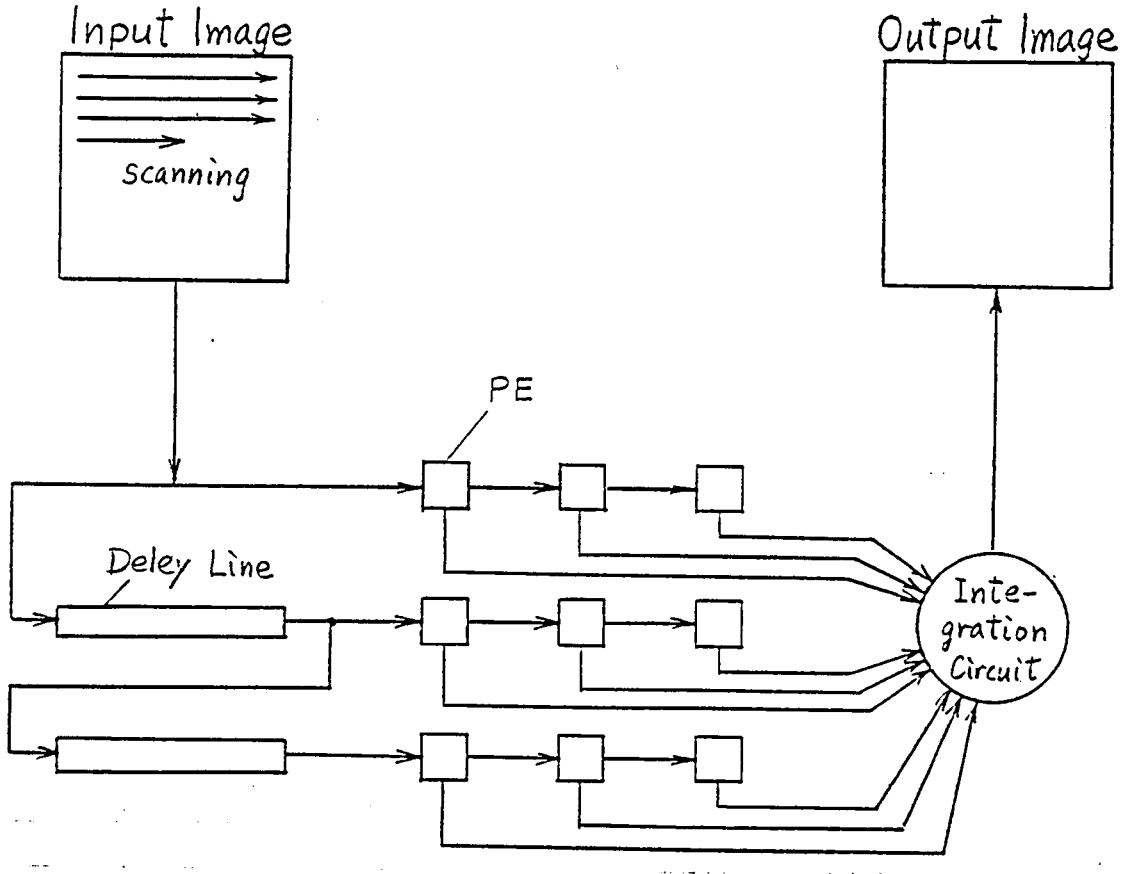


図１．５　局所並列処理を実行する回路の構成例

１．４　高性能画像処理ＬＳＩの研究における基本方針

近年、画像処理技術の研究が活発に行なわれるようになってきているが、その応用は、医療エレクトロニクス・リモートセンシング・半導体製造などの特定の分野に限られている。人間の視覚メカニズムが解明されていない現実を考えると、やむを得ない面もあるが、これ迄に開発されて来た画像処理アルゴリズムすら、その有用性が明らかにされていても、システム構築技術の立ち遅れから、その多くが実用化に至っていないのである。そこで、画像処理技術を一般産業に幅広く普及させることを最大の目的として、高性能画像処理ＬＳＩの研究に取り組んだ。

本研究の推進に当っては、実用化に主眼を置いて、次の三点をその基本方針とした。

（１）高速処理

現在、標準となりつつある工業用テレビ（ＩＴＶ－Ｉndustrial Ｔelevision）カメラは、256×256画素サイズのノンインタレース（非飛び越し走査）画像を、１秒間に６０フレーム生成する。この画像を、生成する速度、つまりＩＴＶカメラの走査速度で実時間処理できるＬＳＩの開発を目指した。カメラから出力される映像信号をそのまま処理できれば、バッファとしての画像メモリが不必要になり、適用分野を拡大することができる。この処理速度は、画素当たりの処理時間が１６７ナノ秒（１ナノ秒は１０億分の１）に相当し、動作周波数は６ＭＨｚとなる。

（２）多機能性

応用分野が異なれば、適用されるアルゴリズムも異なる。そのため、さまざまな分野への応用を目指すためには、多種の画像処理機能を実行できなければならない。特に、今後より高度な処理を実現するためには、２値画像だけでなく、濃淡画像や色彩画像をも扱えることが不可欠である。そこで、２値・濃淡・色彩画像の基本的な演算を、すべて実行できることを目指した。

（３）拡張性

画像処理の機能は多岐にわたるが、すべての機能を網羅することと共に、すべての処理を高速に処理することもまた困難である。特に、画像処理の中でも基本的な演算である局所近傍演算において、演算領域のカーネルサイズの拡張に対応できることは、ＬＳＩの応用範囲を拡げる上で非常に重要である。そこで、開発するＬＳＩのアーキテクチャに柔軟性を持たせて、サイズの異なるカーネルに対処できることとした。

１．５　まとめ

画像処理技術はミニコンピュータの出現により、リモートセンシングと医療エレクトロニクスの分野から実用化が始まった。これは、それまでの大型電子計算機と比べて、ミニコンピュータの性能価格比が数倍優れていたからである。現在、ミニコンピュータよりさらに性能価格比のよいマイクロコンピュータが出現するに及んで、画像処理技術は、一般産業へ広く普及してゆく方向にある。

しかし、マイクロコンピュータの絶対的な処理能力の低さから、実用化は処理の簡単な２値画像に限られ、応用範囲が限定されている。このため、より高度な処理が可能な濃淡画像処理の技術が要望されているが、濃淡画像演算は複雑な演算であり、高速に実行するためには、画像処理専用ＬＳＩの開発が不可欠である。

画像処理用ＬＳＩには二つの流れがある。一つは、画像の全画素を同時に演算する完全並列型であり、もう一つは、１個の出力画素に係る局所画像を並列処理する局所並列型である。完全並列型としては、ＣＬＩＰ－４・ＤＡＰ・ＭＰＰ・ＡＡＰなどが発表されている。１チップには、それぞれ８、１６、８、６４個のＰＥ（演算モジュール）を搭載している。また、局所並列型に属する画像処理用ＬＳＩとしては、米国ヒューズエアクラフト研究所で試作されたテストチップがある。このＬＳＩの主な演算機能は、３×３空間積和演算・ラプラシアン・メディアンフィルタなどである。

本研究においては、画像処理技術を一般産業へ広く普及させることを目的として、①工業用テレビカメラからの映像を実時間で処理できる高速処理、②２値・濃淡・色彩画像の基本的な演算をほとんどすべて実行できる多機能性、③カーネルサイズの拡張に対処できる拡張性、を備えた高性能画像処理ＬＳＩを開発することを基本方針とした。

１．６　第１章の参考文献

[1]長尾　真：画像処理論、コロナ社 (1983).

[2]Nudd, G. R. : Image Understanding Architecture, National Computer Conference, pp.377～390 (1980).

[3]Unger, S. H. : A Computer Oriented toward Special Problems, Proc. IRE, Vol. 46, No. 10, pp.1744～1750 (1958).

[4]坂上　勝彦、木戸出　正継：イメージプロセッサの最近の動向、電子通信学会誌、Vol. 67, No. 1, pp.90～98 (1984).

[5]Duff, M. J. B. : CLIP4 A Large Scale Integrated Circuit Array Parallel Processor, ibid, pp.728～733 (1980).

[6]Hunt, D, J. : The ICL DAP and its Application to Image Processing, edited by Duff, M. J. B. and Levialdi, S., pp.275～282, Academic Press, New York (1981).

[7]Tsoras, J. : The Massively Parallel Processor (MPP) Innovation in High Speed Processor, AIAA Computers in Aerospace Ⅲ Conference, pp.196～201 (1981).

[8]Sudo, T., Nakashima, T., Aoki, M. and Kondoh, H.: An LSI Adaptive Array Processor, IEEE ISSCC Digest of Technical Papers, Vol. 25, pp.122～123 (1982).

[9]Preston, K. Jr. : Cellular Logic Computers for Pattern Recognition, Computer, Vol. 20, No. 1, pp.36～47 (1983).

第２章　画像処理用ＬＳＩ－ＩＳＰの基本アーキテクチャ

２．１　まえがき

画像処理・パターン認識の研究は、近年、ＬＳＩなどのデバイス技術や、並列処理・高速処理などのアーキテクチャ技術の飛躍的な進展に支えられて、実用化の時期を迎えようとしている。特に、ＭＥ(Medical Electronics)やＦＡ(Factory Automation)など、すでに適用の進んでいる分野でも、今後はより高度な処理が要求されるものと考えられる。そのためには、画像の最小構成要素である画素のおのおのに、少なくとも２５６階調程度の濃淡度をもたせた、濃淡画像の高速処理技術を開発してゆかなければならない。しかし、種々の画像演算を高速に処理できるＬＳＩがなかったため、高速・高機能で、かつ安価な画像処理システムの開発は、従来困難であった。そこで、濃淡画像処理技術の実用化を目指して、高速性・拡張性・多機能性を備えた画像処理用ＬＳＩ－ＩＳＰ(Image Signal Processor)を開発した[１，２]。

画像処理において、高速化を実現する並列処理方式は、第１章で述べたように、大きく二つに分類される。一つは、全画像のそれぞれの画素に一つずつＰＥ(Processor Element)を用意して、全画素並列に演算する完全並列型である[３，４]。もう一つは、一つの出力画素を算出するのに用いる入力画素と同数のＰＥを用意して、局所的に並列演算する局所並列型である[５，６]。

完全並列型の場合、処理する画像のそれぞれの画素に、一つずつＰＥを用意するため、演算の並列度は極めて高く、非常に高い処理性能が期待できる。しかし、処理する画像のサイズが大きくなると、必要なＰＥ数は画素数に比例して大きくなり、ハードウェア量は膨大になるという短所がある。

一方、局所並列型の場合、必要なＰＥ数は、処理する画像のサイズと関連なく、常にカーネルを構成する画素数と同数なので、完全並列型と比較して演算の並列度は低いが、ハードウェア規模は非常に小さくなる。なぜなら、通常のカーネル(局所演算領域)の大きさは、３×３ないし５×５程度だからである。また、処理性能については、完全並列型には及ばなくとも、テレビ画像の走査速度で処理することは充分可能である。

つまり、近傍画像演算だけに絞ると、画像処理技術の一般産業への応用において、高い処理性能が得られる完全並列型よりは、小型で安価なシステムが構築できる局所並列型の方が適していると言える。そこで、画像処理技術の実用化を目指す上から、ＩＳＰは局所並列型を指向することにした。

局所並列型の画像処理用ＬＳＩを開発するにあたり、次の三つの点を開発方針とした。

（１）　空間積和演算に代表される局所画像演算を、ビデオレート(１６７ｎｓ／画素)で高速処理できる。

（２）　任意の大きさのカーネルの処理が可能である。

（３）　ＬＳＩの制御信号を変更することにより、２値画像・濃淡画像の種々の基本演算を実行できる。

なぜなら、現在、工業用に最もよく用いられているテレビカメラから採取される画像は、

２５６×２５６画素サイズ程度であり、この画像を実時間で、つまり画像の走査に追随して処理できる速度は、最低限度必要とされること。また、応用目的によって、画像処理に要求される機能やカーネルの大きさは、千差万別だからである。

しかし、実際に応用することを前提とすると、画像処理用ＬＳＩに対して、(1)高速である、(2)拡張性がある、(3)多機能である、という技術的な要素とは違った尺度、ものさしが必要である。それは、価格、つまり経済性である。いかに技術的に優れていても、高価過ぎては実際の応用には適さない。さらに、ＬＳＩ素子そのものは安価であっても、その素子を用いて設計したシステムが高価になるようでは、ＬＳＩ開発の意義を損ってしまう。そこで、システム設計とデバイス設計の両面から、画像処理用ＬＳＩのアーキテクチャを検討した。

本章では、上記三つの開発方針に加え、経済性をも考慮して検討した、画像処理用ＬＳＩ－ＩＳＰの基本アーキテクチャについて述べる[7]。

２．２　基本構想

濃淡画像処理技術を一般産業へ幅広く適用する上から、画像処理用ＬＳＩ－ＩＳＰは局所並列型とした。ここでは、高速・高機能でかつ性能価格比の高い画像処理システムを、構築しうるＬＳＩのアーキテクチャを決定する上での、基本的な構想について検討する。

局所並列型画像処理ＬＳＩの基本アーキテクチャを検討する上で、次式で示すような空間積和演算を題材とした。すなわち、入力画像をｆ(ｘ,ｙ)、出力画像をｇ(ｘ,ｙ)、荷重係数をｗijとするとき、カーネルが４×４の場合の空間積和演算は、次式で表わされる(ただし、参照画素の位置は左上方向へずれる)。図２．１に、４×４空間積和演算の模式図を示す。

$$g(x,y)=\sum_{i=1}^{4}\sum_{j=1}^{4} wij\times f(x+i-1,y+j-1).$$

このような空間積和演算を題材とした理由は、空間積和演算が濃淡画像の最も基本的な局所画像演算であること、および、他の局所画像演算の多くが、積和演算の機能的変形ととらえることができ、同じアーキテクチャで処理できること、などである。

ここで、局所並列型画像処理ＬＳＩのアーキテクチャを検討する上での問題点を整理すると、以下の項目が挙げられる。

（１）　１チップに搭載するＰＥ数。
（２）　１チップ上のＰＥが形成するカーネルの形状。
（３）　画像データと荷重係数のそれぞれのＰＥへの分配方法。
（４）　カーネルの拡張方法。
（５）　ＬＳＩ間のリンケージ方法。
（６）　高速演算方法。
（７）　種々の画像演算の実現方法。

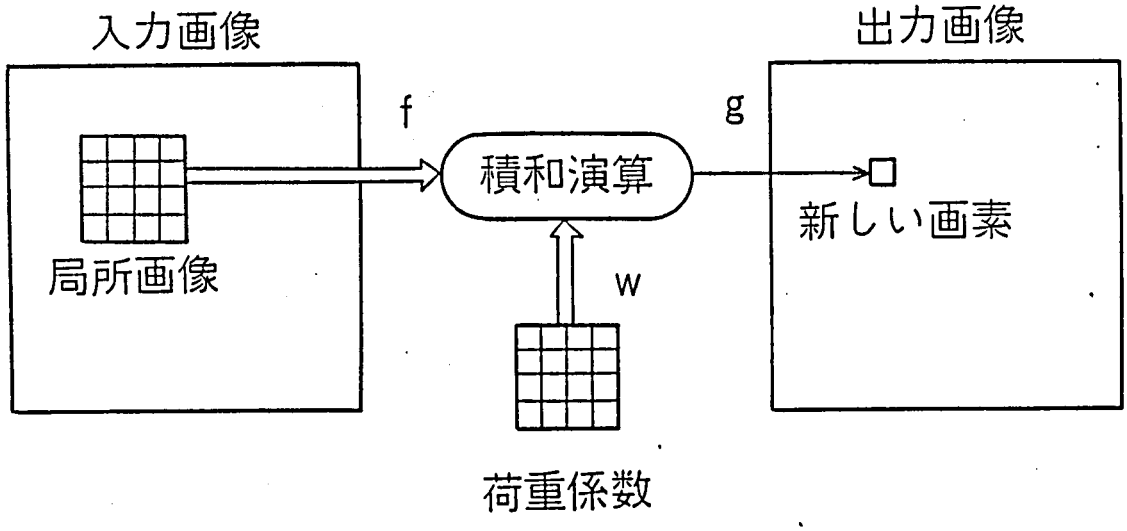


図２．１　４×４空間積和演算

まず、(1)の1チップに搭載するＰＥ数は、理想としては、局所近傍のカーネルを1チップで形成できる数が望ましい。たとえば、カーネルが3×3ならば9個であり、4×4ならば16個である。しかし、画像データおよび荷重係数はともに8ビットが望ましいので、それぞれのＰＥは8ビットの乗算器を持たなければならない。また、(7)の種々の画像演算を実現する上から、それぞれのＰＥには乗算器以外の演算器も必要である。このため、一つのＰＥは約2000ゲートの規模になってしまう。そこで、現状のＬＳＩ技術から、1チップ4ＰＥが適当であると判断した。この結果、図2．1に示した4×4空間積和演算を、局所並列で処理するには、4個のＬＳＩが必要となる。

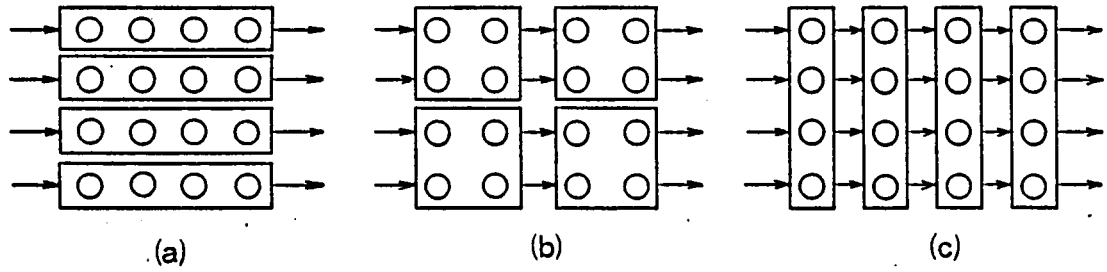
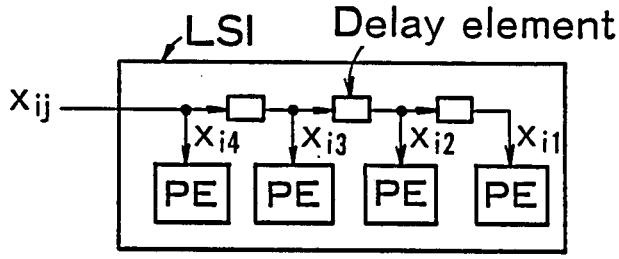


図2．2　カーネルの分割方法

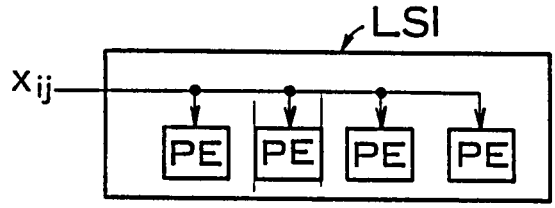
(2)は、上の結論から、4×4のカーネルを並列処理する16個のＰＥを、どのように4チップに分割するのか、という問題となる。これには、図2．2に示す三つの方法が考えられる。図2．2において、○印はＰＥを示し、矢印は画像データの転送方向を示す。ここで考慮しなければならないのは、ＬＳＩの端子数である。端子数が増えると、チップやパッケージが大きくなり、歩留りの低下、コストの上昇、実装密度の効率悪化などにつながる。しかし、端子数を無理に制限すると、画像データの供給が時分割になり、処理速度が低下する。これらの点を考慮して、端子数が少なくても高速処理が可能な2．2(a)の分割方法を採用した。図2．2(a)の方法によると、画像データの入力および出力は、それぞれ8本の端子ですむ。なお出力端子は、カーネルを拡張する際に、別のＬＳＩの入力端子に接続される。

図2．2(a)の分割を前提とすると、(3)の画像データの分配方法および荷重係数の供給方法には、図2．3および図2．4の二つの方式が考えられる。それぞれどちらの方式を選択しても、データの入力方法やＰＥの出力結果の演算の方法などの工夫により、空間積和演算を実行できる。しかし、積和演算以外の画像演算の実行や、端子数の削減などを考慮して、画像データの分配には図2．3(a)の遅延方式を、荷重係数の分配には図2．4(a)の内蔵方式を採用した。これに関しては第2．3節で詳しく述べる。

(4)のカーネルの拡張に対しては、次の二つのアプローチの方法がある。一つは、カーネルのサイズに合わせてＰＥを用意する方法である。もう一つは、少数のＰＥを時分割に使用する方法である。画像処理システムを構築する場合、処理速度を第一の要件とするものと、より小型化を指向するものとがある。二つのアプローチのうちの前者の方法によると、高速処理が可能で、後者の方法によると、システムの超小型化が達成できる。ＩＳＰでは、それぞれの長所を活かすため、両方法を実現しうるアーキテクチャを考案した。こ

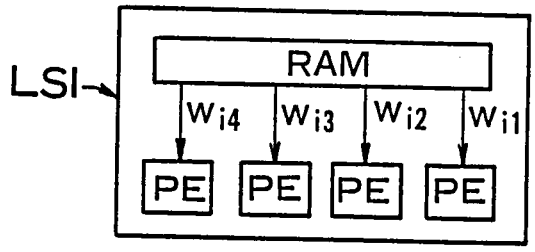


(a) Delayed distribution

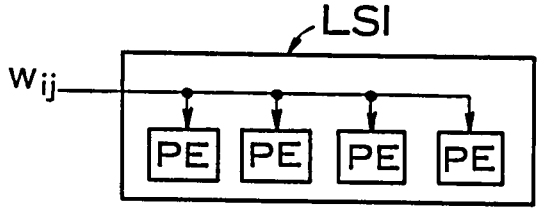


(b) Common distribution

図２．３　画像データ分配方法



(a) Internal supply



(b) External supply

図２．４　荷重係数供給方式

れに関しては第２．４節で詳しく述べる。

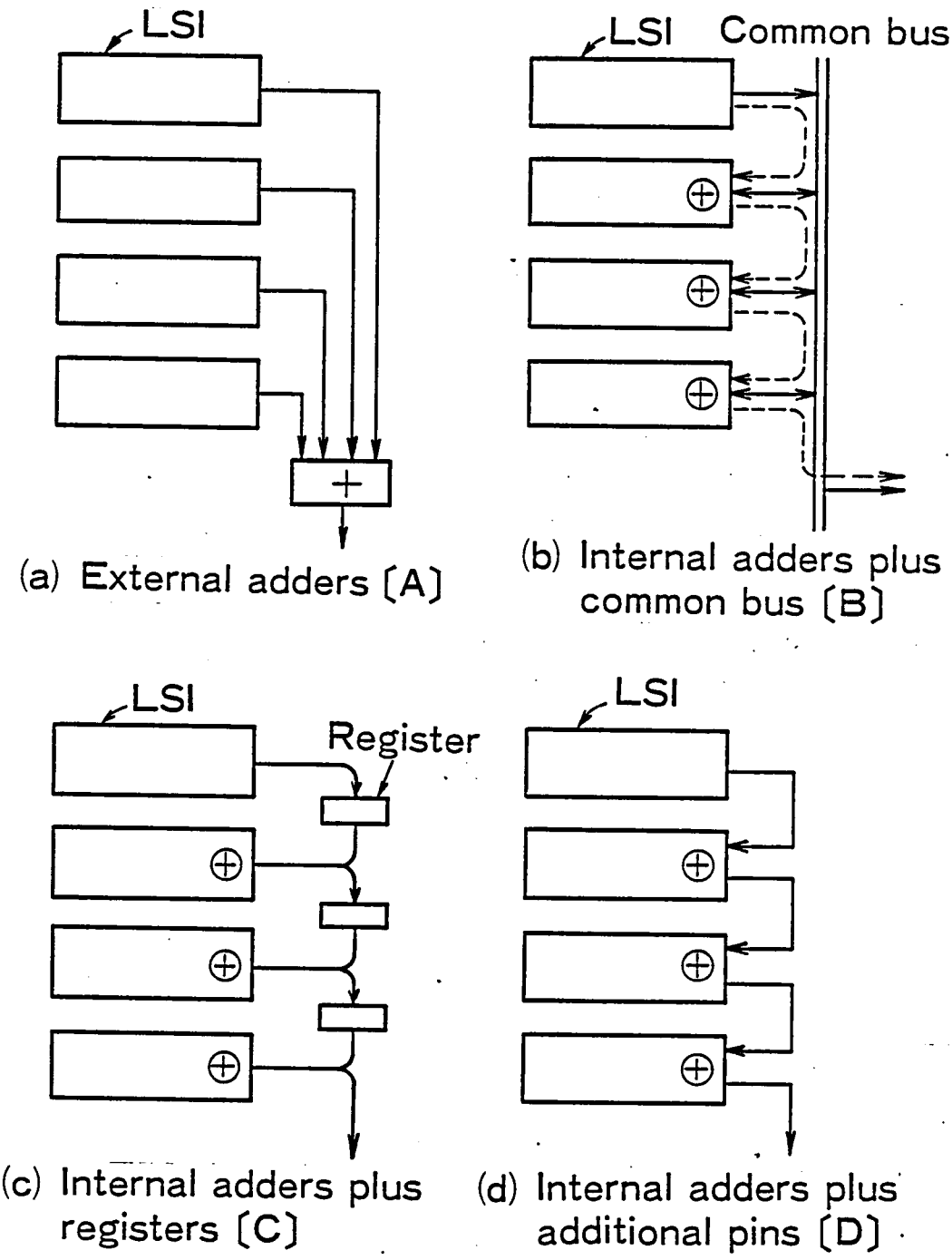
複数個のＬＳＩを用いる場合、(5)のＬＳＩ間のリンケージ方法は、システムの性能を決定する上での大きな要素である。４個のＬＳＩを用いる場合を例にとると、リンケージ方法として、図２．５に示す四つの方法が考えられる。図２．５(a)の方式は、高速に処理できるが、ＬＳＩ外部に加算器が不可欠である。図２．５(b)の方式は、ＬＳＩ外部に加算器は必要ないが、ＬＳＩ間のデータ転送がボトルネックとなり、図２．５(a)と同じ処理速度を保つには、データ転送速度を図２．５(a)の４倍にしなければならない。図２．５(c)の方式は、図２．５(b)の方式より転送効率はよく、データ転送速度を図２．５(a)の２倍でよいが、ＬＳＩ外部にレジスタを必要とする。図２．５(d)の方式は、図２．５(c)よりさらに転送効率がよく、図２．５(a)と同じデータ転送速度で高速処理できるが、端子数が増加する。以上の考察を踏まえ、ＩＳＰには拡張の容易さと高速性を重視して、外部回路が不必要で高い処理性能が得られる図２．５(d)の方式を採用した。端子数の増加には、プログラマブルな制御レジスタをＬＳＩ内部に内蔵することなどにより、制御信号用の端子をできるだけ少なくすることで対応した。なお、リンケージデータは１６ビット幅としたので、リンケージデータ用に合計３２本の端子が必要である。

(6)の高速演算方法については、２５６×２５６画素から成るノンインタレースの濃淡画像を、ビデオレートで高速処理できることを基本とした。これは、１画素当たりの処理時間が、１６７ｎｓになることを意味する。この高速処理を実現するため、ＩＳＰには、並列処理に加えてパイプライン処理方式を採用した。ＩＳＰにおいては、ＬＳＩ内だけでなく、複数個のＬＳＩを用いた時、ＬＳＩ間においてもパイプライン処理が可能となるアーキテクチャとした。ＬＳＩ間のパイプライン処理方式については、第２．５節で詳しく述べる。

ＬＳＩ内のパイプライン処理方式は、(7)の種々の画像演算の実現と合わせて検討し、ＩＳＰでは、プログラマブル制御レジスタによる構成制御を基本とした。つまり、演算実行前に、プログラマブル制御レジスタの内容を変更することにより、パイプラインを構成するそれぞれの演算回路の機能や、画像データや荷重係数などの流れを決定し、それらの組み合わせにより種々の画像演算が実現される。この種々の画像演算の実現については、第３章で詳しく述べる。

以上を要約すると、ＩＳＰのアーキテクチャを形成するための基本構想は、次のようにまとめられる。

（１） １チップに４個のＰＥを搭載する。
（２） ４個のＰＥは１×４の構成とする。
（３） 画像データは遅延回路を介して順次ＰＥに分配され、荷重係数は内蔵ＲＡＭからＰＥに供給される。
（４） カーネルはＬＳＩの追加もしくは時分割使用、いずれによっても拡張される。
（５） ＬＳＩ間のリンケージには、専用の入出力端子と演算器を用意する。
（６） ＬＳＩ内およびＬＳＩ間に、マシンサイクルが１６７ｎｓのパイプライン処理を構築する。
（７） プログラマブル制御レジスタによる構成制御を用いる。



第２．５　ＬＳＩ間リンケージ方式

２．３　ＰＥへのデータ供給方式

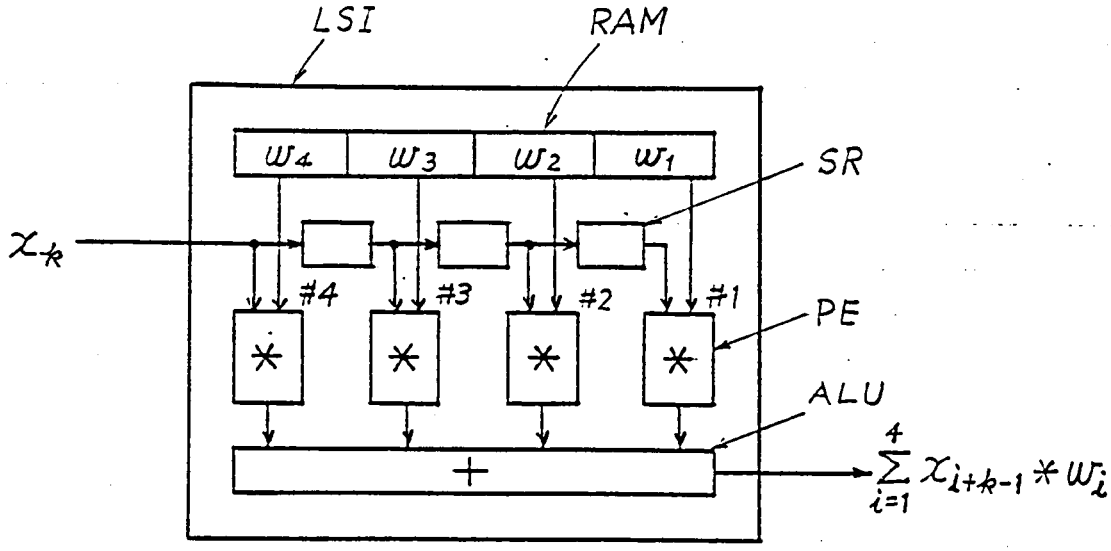
前節では、局所並列型画像処理ＬＳＩのアーキテクチャを形成するための、基本構想を明らかにしたが、本節ではこのうち、ＰＥへの画像データの分配方法と、ＰＥへの荷重係数の供給方法について、①積和演算を題材とする、②１チップに４個のＰＥを搭載する、③４個のＰＥは１×４の構成とする、ことを前提として詳しく論ずる。

上記の①、②、③を前提とすると、ＰＥへの画像データの分配方法および荷重係数の供給方法には、前節で述べたように、それぞれ二つの方式が考えられる。画像データに対しては、図２．３(ａ)の遅延分配方式と同図(ｂ)の共通分配方式があり、荷重係数については、図２．４(ａ)の内部供給方式と同図(ｂ)の外部供給方式がある。その結果、それぞれの組合せにより、４通りのＰＥデータ供給方式が考えられる。以下、これら４通りのそれぞれについて、積和演算を実行する際のＬＳＩの内部構成と、その長所および短所について論ずる。

(１)　遅延分配方式 ＋ 内部供給方式

本方式により、１×４積和演算を実行する際のＬＳＩの内部構成は、図２．６に示すようになる。またその時のデータフローは、図２．７のように示される（ただし、図２．６のＳＲ（Shift Register）、ＰＥ、ＡＬＵ（Arithmetic Logic Unit）は、すべて１マシンサイクルで動作し、それぞれが一つのパイプラインステージを形成しているものとする)。

図２．６において、画像データは、１マシンサイクルに１画素ずつ、x_1, x_2, x_3, ．．．，と順次入力される。入力された画像データｘｋは、ＳＲを介してＰＥに分配され、内蔵ＲＡＭから供給される荷重係数ｗｉと、ＰＥで掛け合わされた後、１６ビット４入力ＡＬＵで足し合わされる。足し合わされて求められた１×４積和演算の結果は、ＬＳＩ外部に出力される。この時、ＰＥおよびＡＬＵが、それぞれ１段のパイプラインステージを形



第２．６　遅延分配方式と内部供給方式によるＬＳＩの内部構成

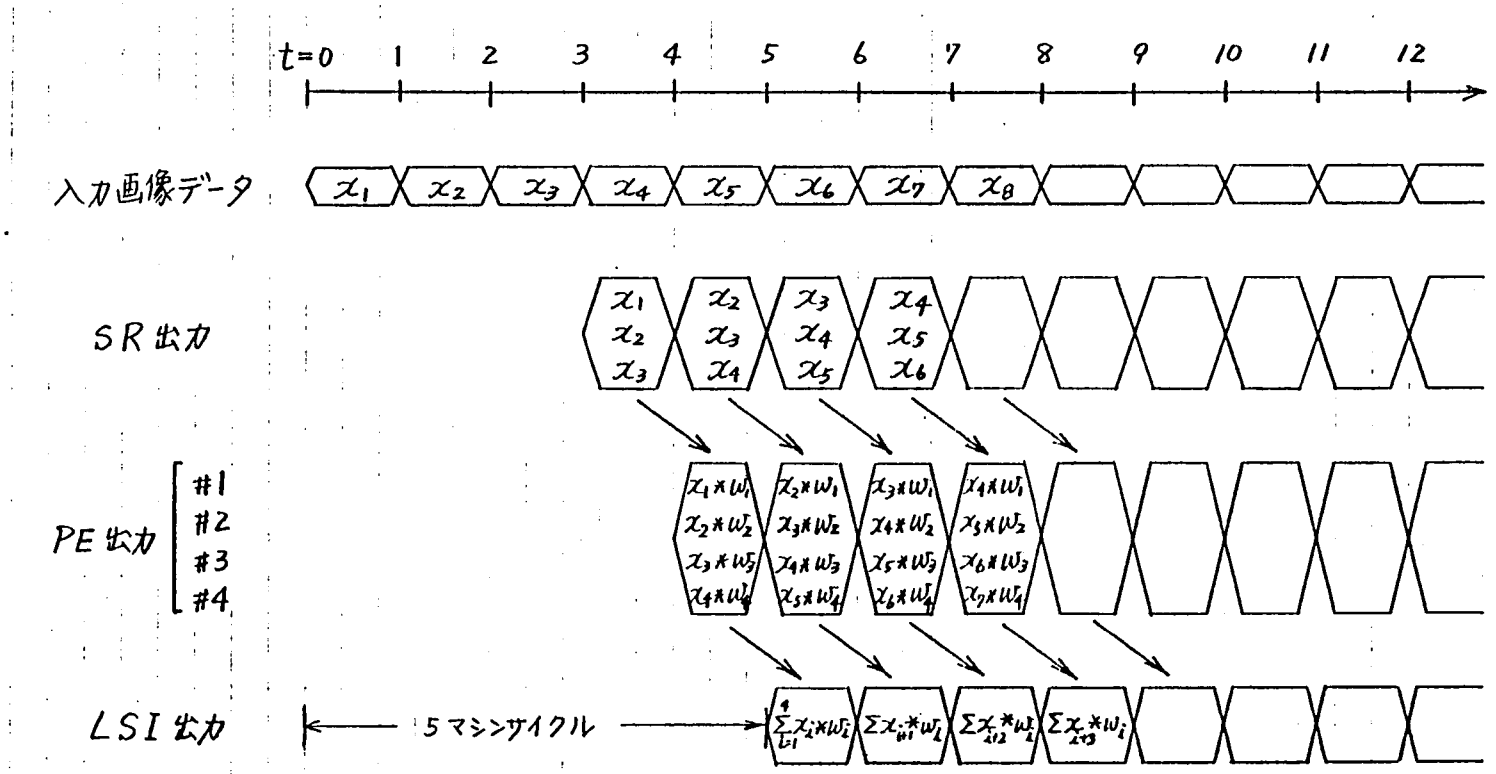


図２．７　遅延分配方式と内部供給方式の構成による
１×４積和演算実行時のデータフロー

成するので、演算結果は、画像データが入力された５マシンサイクル後から、順次出力されることになる。つまりＬＳＩは、５段のパイプライン演算器として動作する。

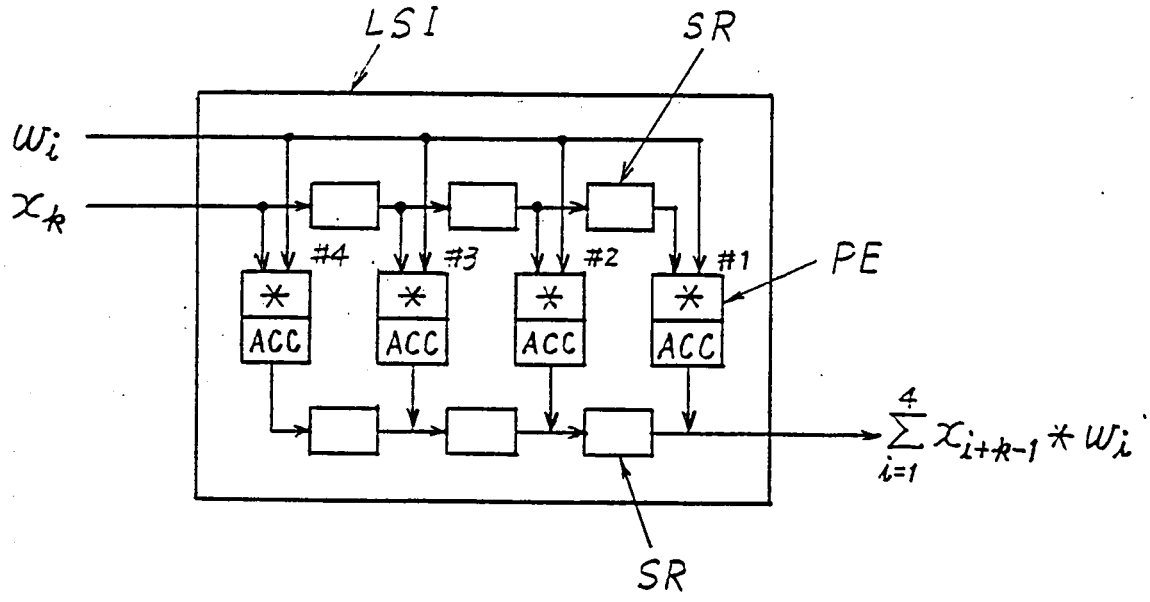
本方式では、８ビット×８ビット乗算器４個、１６ビット４入力加算器１個（もしくは１６ビット２入力加算器３個）、ＳＲ３個および４語分（３２ビット）のＲＡＭ（Random Access Memory）によりＬＳＩを構成できる上、内部構成が単純なパイプラインステージを形成するため、論理構成も単純になり、制御も容易になるという利点がある。

（２） 遅延分配方式 ＋ 外部供給方式

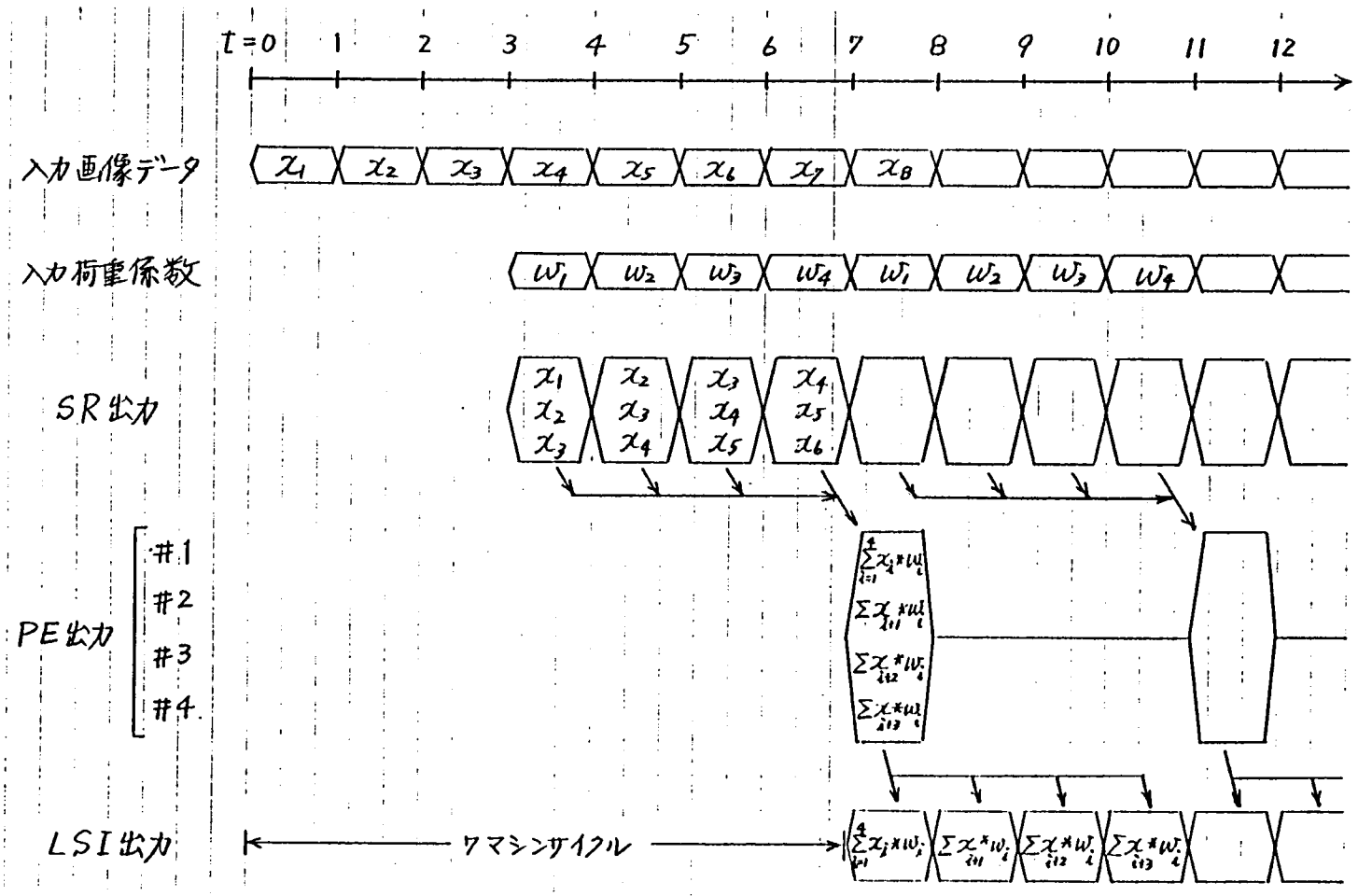
本方式により１×４積和演算を実行する際のＬＳＩの内部構成は、図２．８に示すようになり、またその時のデータフローは、図２．９のように示される。

図２．８において、画像データは一方の入力端子に、１マシンサイクルに１画素ずつ、x_1, x_2, x_3, ... , と順次入力されるが、１×４積和演算を実行するには、もう一方の入力端子に荷重係数（w_1, w_2, w_3, w_4）が、サイクリックに入力されなければならない。なぜなら、（１）の方式と異なって、それぞれのＰＥが同時にしかし独立して、１×４積和演算を実行するからである。つまり、４個のＰＥが、４マシンサイクルで四つの１×４積和演算を実行することになる。これら４個の演算結果は、ＰＥが次の積和演算を実行する４マシンサイクルの間に、ＬＳＩ外部に出力される。このため、図２．９からも知れるように、画像データの入力と演算結果の出力だけをみる限り、ＬＳＩは７段のパイプライン演算器とみなせる（ただし、ＰＥにおける乗算と加算は、１マシンサイクルで実行されるものとする）。

本方式を（１）の方式と比較すると、まず荷重係数記憶用の３２ビットＲＡＭと、１６ビット４入力ＡＬＵ（１６ビット２入力ＡＬＵ３個相当）が必要でなくなる。しかし、荷重係数入力用の端子８本や、演算結果出力用のＳＲ３個が必要となるほか、それぞれのＰＥに１６ビットＡＣＣ（Accumulator）が必要となる。そのため、ＬＳＩだけでみたとして



第２．８　遅延分配方式と外部供給方式によるＬＳＩの内部構成



第２．９　遅延分配方式と外部供給方式の構成による
１×４積和演算実行時のデータフロー

も、ハードウェア規模、チップ面積が大きくなる上に、ＡＣＣのための制御の複雑さが加わるため、(1)の方式より優れているとはいえない。

(3) 共通分配方式 + 内部供給方式

本方式により１×４積和演算を実行する際のＬＳＩの内部構成は、単純には図２．１０に示すようになる。これは、図２．８の構成に内蔵ＲＡＭを設け、荷重係数をＲＡＭからＰＥへ供給するものである。そのため、図２．１０の構成のデータフローは、図２．８の構成のデータフローと同じになる。

図２．１０の構成は、(2)の方式による図２．８の構成と比べると、４語の荷重係数記憶用に３２ビットのＲＡＭが必要となるが、入力端子８本を削除できる。入力端子の削除は、ボンディングパッド、入力保護回路、入力バッファ回路の削除を意味し、３２ビットのＲＡＭを新たに追加しても、チップ面積の上からは、大きな効果が期待できる。

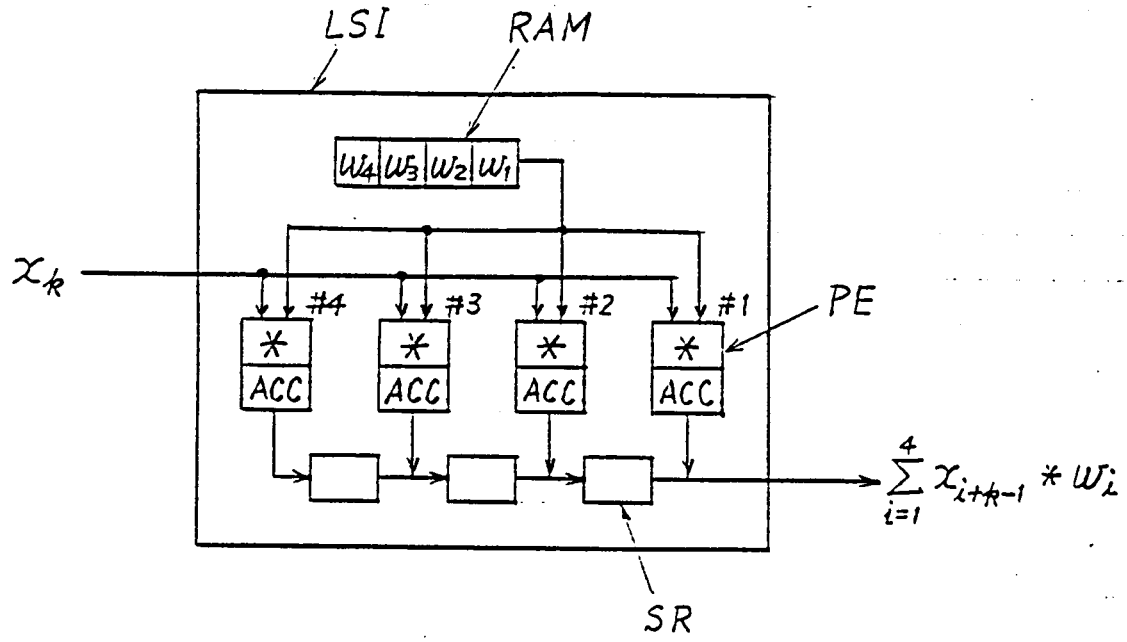
図２．１０の構成を改善して、ハードウェア規模の縮小を図ったのが図２．１１の構成である。図２．１１では、画像データは４個のＰＥに共通して与えられるが、荷重係数は閉ループを構成する４個のＳＲに記憶されている。４個のＰＥはそれぞれ４マシンサイクルで１×４積和演算を実行するが、図２．１２に示すように、それぞれのＰＥで演算サイクルを一つずつずらすことにより、みかけ上４段のパイプライン処理を実現することができる（ただし、ＰＥにおいて乗算と加算が１マシンサイクルで実行できるものとする)。

図２．１１の構成を、(1)の方式による図２．６の構成と比較すると、ハードウェア規模上は、３２ビットＲＡＭを１６ビットＡＣＣで置き換えた程度の差しかなく、ほぼ同規模といえる。しかし、制御の面からみると、図２．１１の構成は、それぞれのＰＥでＡＣＣの制御を個別に行なわねばならない上に、ＬＳＩ外部への出力制御が必要なことから、図２．６の構成より、はるかに複雑になるといえる。

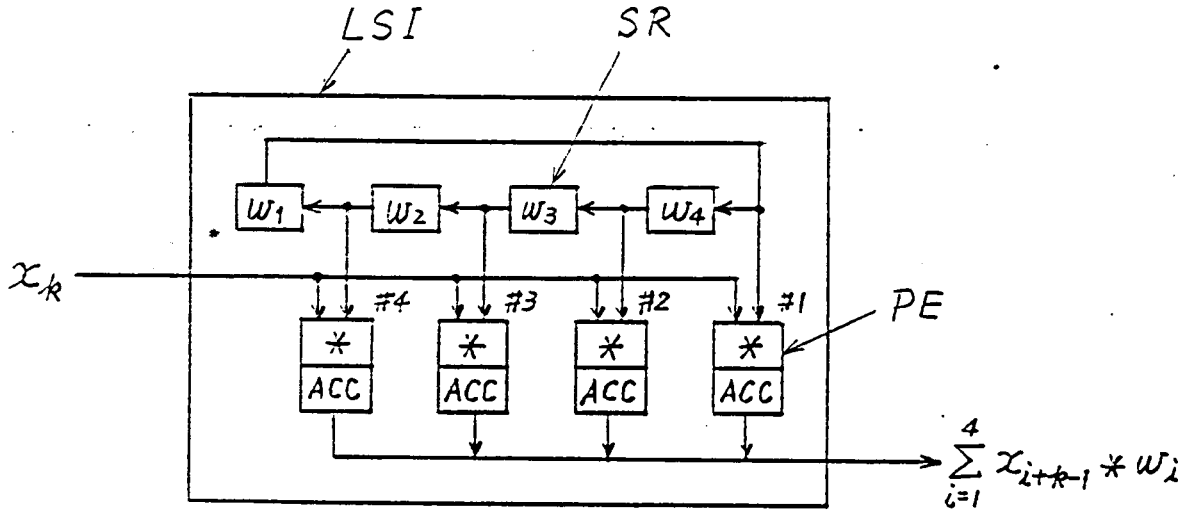
(4) 共通分配方式 + 外部供給方式

本方式により１×４積和演算を実行する際のＬＳＩの内部構成は、図２．１３に示すようになる。図２．１３では図２．８と異なり、荷重係数はそれぞれのＰＥに、ＳＲを介して供給されている。これは、画像データがそれぞれのＰＥに共通に与えられているため、荷重係数に遅延を持たせなければ、効率良く積和演算が実行できないためである。すなわち、図２．１３の構成は、図２．１１において、荷重係数をサイクリックに用いる代わりに、外部から入力しているに過ぎないのである。もちろん、荷重係数は、w_1, w_2, w_3, w_4, w_1, w_2, ... , と繰り返し入力しなければならない。この時のデータフローは、荷重係数の繰り返し入力を除けば、図２．１１の構成のもの（図２．１２）と同じになる。

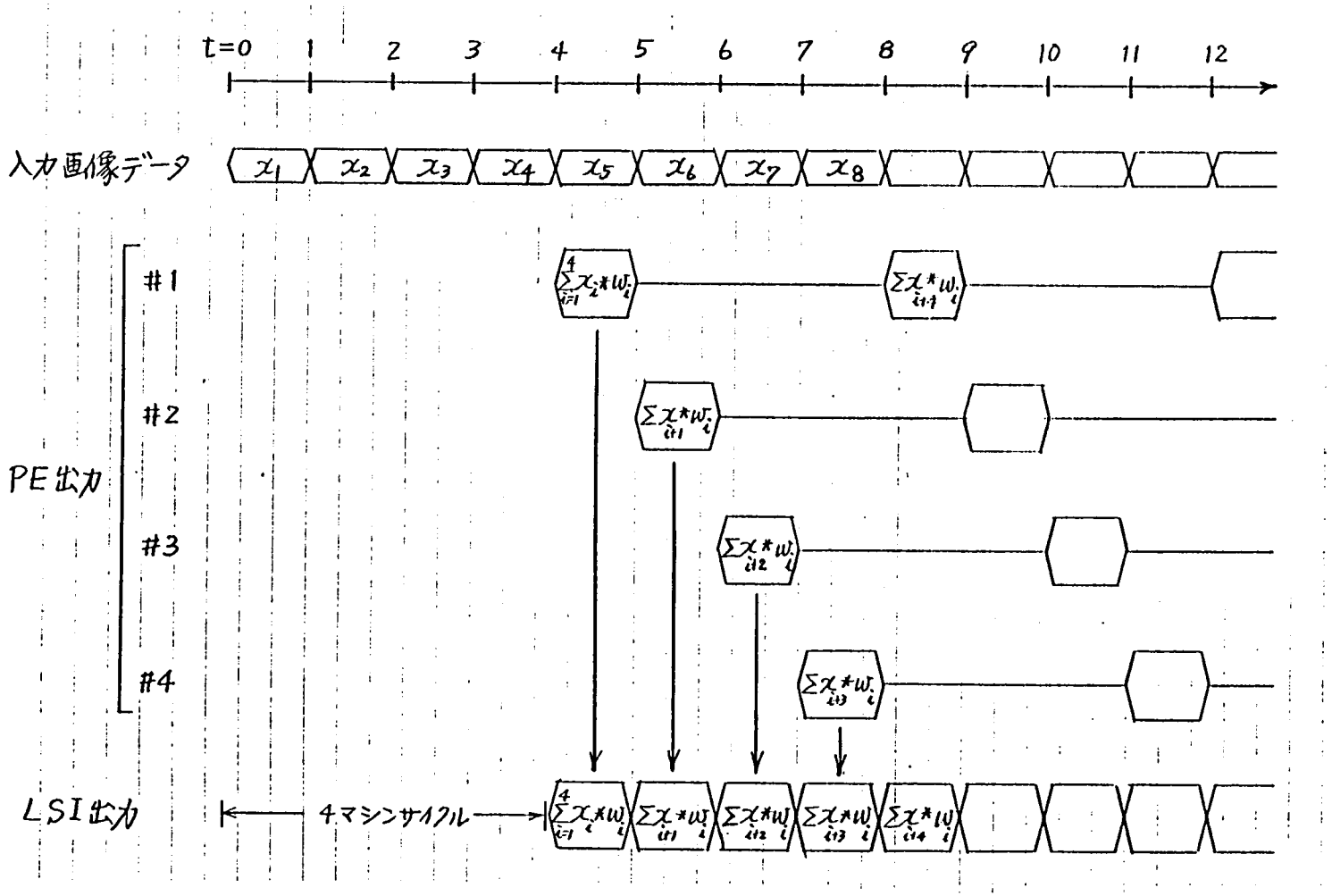
この方式は、図２．１１の方式と比べ、荷重係数入力用の端子８本が増えるだけ、チップ面積の上から不利である。



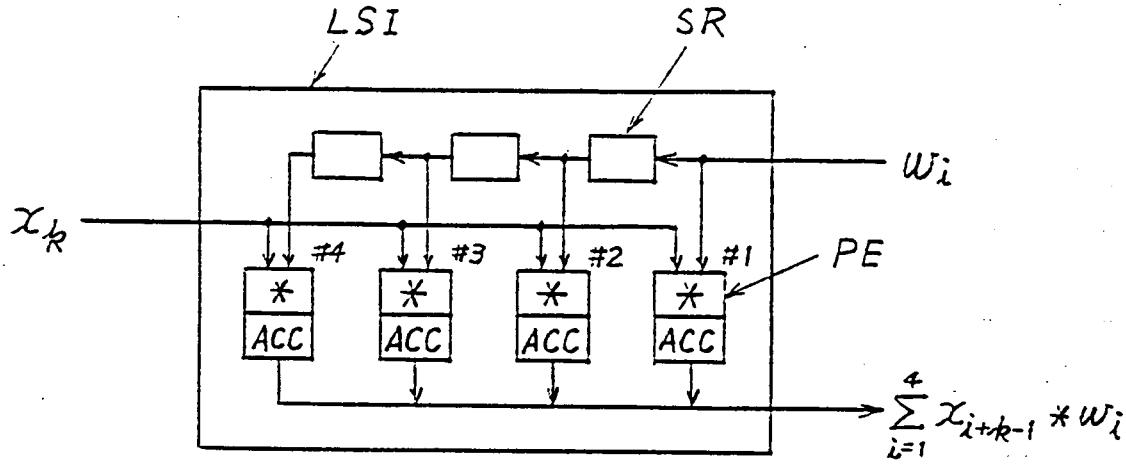
第２．１０　共通分配方式と内部供給方式（ＲＡＭ内蔵）
によるＬＳＩの内部構成



第２．１１　共通分配方式と内部供給方式（ＳＲ使用）
によるＬＳＩの内部構成



第２．１２　共通分配方式と内部供給方式（ＳＲ使用）による
１×４積和演算実行時のデータフロー



第２．１３　共通分配方式と外部供給方式によるＬＳＩの内部構成

以上、四つの方式を検討した結果、入力端子が少なく、ハードウェア規模の小さい、さらに制御の容易な(１)の方式（画像データは遅延分配方式に、荷重係数は内部供給方式による方式）を採用することにした。さらにこの方式は、他の方式に比べると、演算回路構成が単純であるので、積和演算以外の画像処理演算（例えば、第３章で述べる各種の１次微分演算）への適用が容易であるという利点がある。

２．４　高速化と小型化

画像処理システムを構築する場合、何よりも高速化を達成したいという要望と、高速化よりもむしろ小型化を優先させたいという要望がある。第２．２節で述べたように、ＩＳＰでは両方の要望を満たすようなアーキテクチャとした。たとえば、４×４空間積和演算を実行する場合、ＩＳＰを４個用いるとビデオレートで高速処理でき、またＩＳＰ１個でも、ビデオレートの 1/4 の速度で処理できる。

ここでは、二つの方法によるカーネル拡張方式と、それらを可能にする画像走査方式について、さらにそれらを踏まえた、高速化および小型化に適するＩＳＰのアーキテクチャについて述べる。

２．４．１　カーネル拡張方式

ＩＳＰにおいては、二つの方法によりカーネルを拡張できる。一つはＰＥを増やす方法であり、もう一つはＰＥを時分割に使用する方法である。前者をＰＥ増設方式と、後者をＰＥ節約方式と呼ぶことにする。

ＰＥ増設方式では、カーネルを構成する画素数と同数のＰＥを用いるため、カーネルサイズに合わせてＩＳＰを追加することになる。ＩＳＰ１個では１×４のカーネルが処理できるので、４×４のカーネルに対しては４個のＩＳＰ、８×８のカーネルに対しては１６個のＩＳＰが必要になる。この方式ではカーネル全体を並列処理できるので、カーネルの大きさにかかわらず、ビデオレートの処理が可能である。しかし、カーネルを切り出すために周辺回路が必要であり、カーネルが大きくなると必要なＩＳＰ数は増加する。なお、３×３のカーネルを形成する場合は、３個のＩＳＰを用い、ＩＳＰ内のＰＥをそれぞれ３個ずつ動作させる。また、５×５のカーネルの場合、必要なＩＳＰの数は１０個で、５×８のマトリックスを形成する４０個のＰＥの内の２５個を動作させることになる。

これに対して、ＰＥ節約方式ではＰＥを時分割で使用するため、時分割数をｎとすると、原理的には１個のＰＥでｎ×４のカーネルを処理できる。この時の処理速度は、ＰＥ増設方式と比較すると 1/n に低下する。実際には、集積度とチップサイズの問題からｎ＝４としたため、ＩＳＰ１個で４×４までのカーネルが処理できる。８×８のカーネルに対しては、ＩＳＰ４個で処理できる。つまり、処理速度の低下を甘受するならば、少数のＩＳＰでコンパクトなシステムが構築できる。また、次項で述べるスティック走査方式を用いると、カーネル切り出しのための回路も不必要になり。システムをさらに小型化できる。なお、３×３のカーネルを処理する場合、１個のＩＳＰの中の３個のＰＥを用い、時分割数３で対応できる。また、５×５のカーネルに対しては、４個のＩＳＰの中の１０個のＰＥを用い、たとえば２個のＩＳＰはＰＥを４個用いて時分割数３で、他の２個のＩＳＰはＰＥを１個のみ用いて時分割数２相当（時分割数３ならば１サイクルはアイドル状態）で動作させて対応する。

一般産業応用における実際の画像処理システムにおいては、ビデオレートの高速処理が必要な場合と、高速処理よりはむしろシステムを小型化したい場合がある。前者にはＰＥ増設方式を、後者にはＰＥ節約方式を適用すればよい。

２．４．２　画像走査方式

前項で述べた二つのカーネル拡張方式を比べると、画像データの入力方式が同じにならないことは容易に知れる。ＩＳＰのアーキテクチャを論ずる前に、ここでは、それぞれのカーネル拡張方式に適した画像走査方式について考察する。

（１）　ラスタ走査方式（Ｒaster Ｓcanning）

ラスタ走査方式は、通常のテレビ画像の走査方式であり、二つの走査方向から成る。主走査方向は左から右であり、副走査方向は上から下である。図２．１４に示した１０×１０画素から成る小さな画像を例にとると、ノンインタレース画像の場合、それぞれの画素は図示した順に、①から(100)まで順次走査される。

ＰＥ増設方式によりカーネルを拡張すると、ラスタ走査方式に従って走査された画像データを処理できるため、テレビ画像を実時間処理することが可能になる。

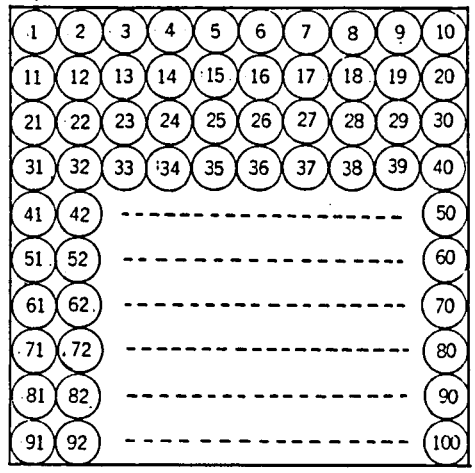


図２．１４　10×10画素から成る画像

（２）　スティック走査方式（Ｓtick Ｓcanning）

ＰＥ節約方式を用いてカーネルを拡張した場合は、ラスタ走査方式によって走査された画像データを、ＩＳＰに入力することはできない。なぜなら、カーネルを形成する画素データのすべてが、ＩＳＰ内に一時記憶されなければならないからである。そこで、ＰＥ節約方式に適したスティック走査方式を、新たに提案する。

スティック走査方式は、ＲＡＭから画像データを読み出す時に実現できる走査方式で、三つの走査方向から成る。主走査方向は上から下、副走査方向は左から右、そして副々走査方向は上から下である。ここで、主走査方向において走査される画素の集合をスティック(stick)と呼び、一つのスティックに含まれる画素数をスティック長(stick length)と定義する。つまりラスタ走査は、スティック長が１の特殊なスティック走査となる。

再び図２．１４の小画像を例にとると、スティック長が４のスティック走査において、それぞれの画素は次の順に走査される(○内の数字は画素＃(番号)を示す)。

①, ⑪, ㉑, ㉛, ②, ⑫, ㉒, ㉜, ③, ⑬, ㉓, ㉝,
④, − − − → ⑩, ⑳, ㉚, ㊵, ⑪, ㉑, ㉛, ㊶, ⑫,
㉒, ㉜, ㊷, ⑬, − − − → ⑳, ㉚, ㊵, ㊿, ㉑, ㉛,
㊶, ㊿, ㉒, − − − → ㉚, ㊵, ㊿, ⑥⓪, − − − → ㊵,
㊿, ⑥⓪, ⑦⓪, − − − → ⑦⓪, ⑧⓪, ⑨⓪, ①⓪⓪.

これから分かるように、スティック走査においては、画像の上下数行を除くと、それぞれの画素はスティック長と同じ回数だけ走査される。

２．４．３　高速化向け構成と小型化向け構成

ＰＥ増設方式によるカーネルの拡張と、ラスタ走査方式による画像データの入力を前提とすると、高速処理に適するＩＳＰのアーキテクチャが固まる。図２．１５にその構成を示す。図２．１５において、画像データは、遅延回路で１段ずつ遅延が加えられて、それぞれのＰＥに分配される。またＲＡＭには４個の荷重係数が記憶され、一つずつそれぞれのＰＥに供給される。

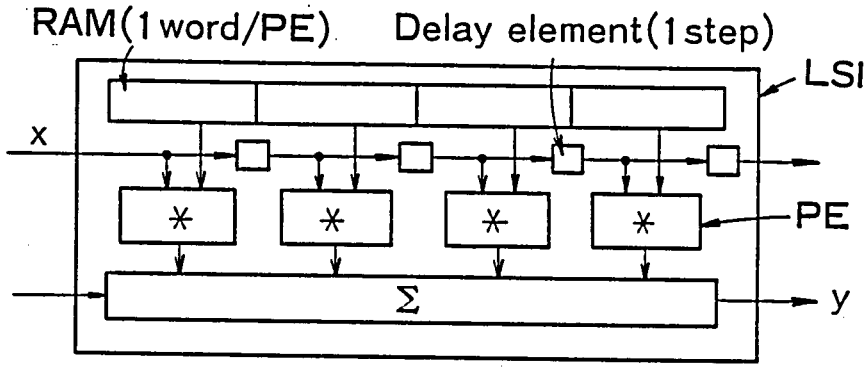


図２．１５　高速処理に適するＩＳＰの構成

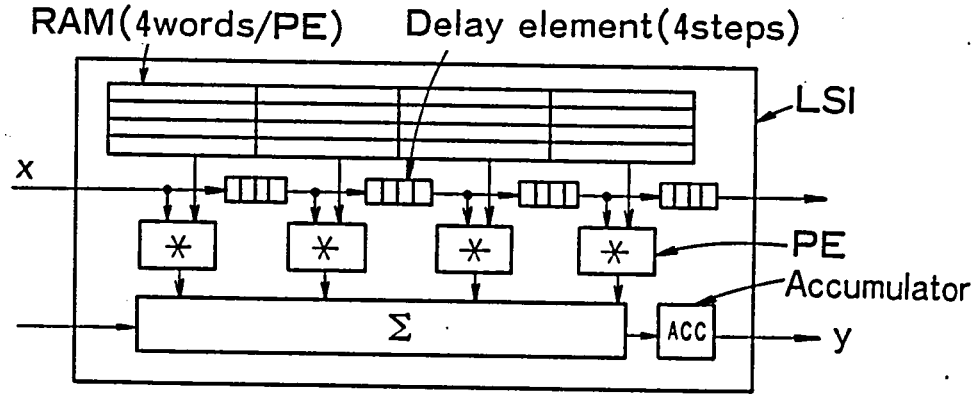


図２．１６　小型化システムの適するＩＳＰの構成

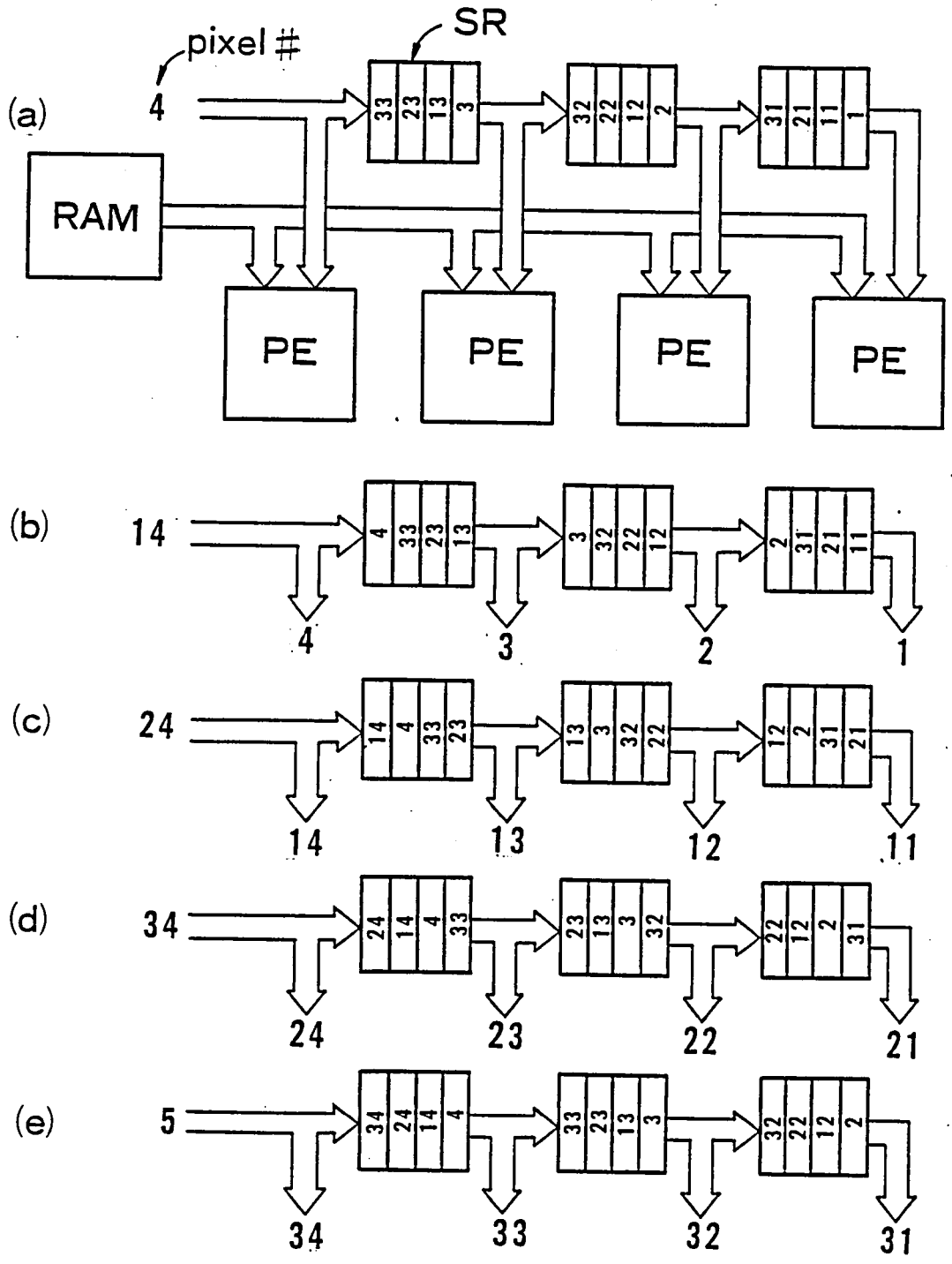


図２．１７　スティック長が４のスティック走査
における画像データ処理

一方、ＰＥ節約方式によるカーネルの拡張と、スティック走査方式による画像データの入力を前提とすると、ＩＳＰのアーキテクチャは図２．１６の構成になる。図２．１６は、次の三つの点で図２．１５と異なる。

（１） それぞれの遅延回路が４段である。

（２） ＲＡＭの容量が４倍である。

（３） アキューミュレータが設けられている。

遅延回路をそれぞれ４段としたのは、４×４のカーネルを形成する画素データを、一時記憶するためである。ＲＡＭの容量を４倍にしたのは、４×４のカーネルに必要な荷重係数を記憶するためである。またアキューミュレータは、時分割に求められる４個の部分演算結果を統合するために設けた。

図２．１６において、前項で述べたスティック長が４のスティック走査を用いた場合の、画素データの処理方法を図２．１７に示す。

図２．１７において、それぞれの遅延回路の段数は、スティック長と同じ４となっている。図２．１７(ａ)は、図２．１４の小画像を入力として、画素＃３３まで入力された状態を示している。画素＃４が、次に入力される画素データである。図２．１７(ｂ)は、図２．１７(ａ)から１マシンサイクル後の、画素＃４が入力された状態である。この時、画素＃１、＃２、＃３、＃４のデータが、対応するＰＥに供給されている。図２．１７(ｃ)、(ｄ)、(ｅ)は、図２．１７(ｂ)から、それぞれ１マシンサイクルずつ経過した時点の状態である。図２．１７から分かるように、(ａ)から(ｅ)までの４マシンサイクルの間に、４×４のカーネルを形成する画素データがＰＥに供給されている。また、図２．１７の(ａ)と(ｅ)を比べると、次の４マシンサイクルでは、右へ１画素シフトした４×４のカーネルの画素データが、ＰＥに供給されることが分かる。つまり４×４のカーネルは、４マシンサイクル毎に１画素ずつ右へ移動してゆく。この時、カーネル切り出しのための回路は不必要である。

一方、空間積和演算における荷重係数は、ＩＳＰ内蔵のＲＡＭからＰＥに供給されるが、４マシンサイクルの間に、１６個の荷重係数がＰＥに供給されなければならない。そこで、ＲＡＭは１アドレス４ワードの並列構成とし、端子から直接アドレスする方式とした。

このように、アーキテクチャを若干変更することにより、高速化にも小型化にも対応することが可能である。ＩＳＰでは、プログラマブル制御レジスタによる構成制御により、いずれのアーキテクチャもとれるようにした。

２．５　ＬＳＩ間パイプライン処理

産業用テレビカメラとして標準になりつつある固体撮像カメラは、２５６×２５６画素程度から成るノンインタレース画像を、１秒間に６０フレーム生成する。つまり、１６．７ｍｓ毎に、新しい画像が生成されることになる。この画像を実時間で処理するには、無効領域を考慮して、１６７ｎｓ以内に一つの画素を処理しなければならない。これを実現するために、ＬＳＩ内部の演算は、すべてパイプライン処理とした。さらに、複数個のＬＳＩを用いる場合にも対処するため、ＬＳＩ間のリンケージ演算もパイプライン処理とした。しかし、ラスタ走査やスティック走査などの画像入力方式によらず、複数個のＬＳＩを用いてＬＳＩ間のパイプライン処理をする場合、オペランド間に時間的な歪みが生ずる。この問題を、歪み補正バッファを設けることで解決した。

ＬＳＩ間の演算にパイプライン処理を導入すると、ＬＳＩ間パイプラインの段数だけ演算データが遅延して、リンケージ演算のオペランド間に、時間的な歪みが発生する。この時間歪みを補正するためには、それぞれのＬＳＩで、ＬＳＩ間パイプラインの段数を考慮して、リンケージ用演算回路に演算データを与えなければならない。つまり、それぞれのＬＳＩで、演算データに異なる遅延を与えなければならない。この問題を解決するため、ＩＳＰでは、画像データの入力バッファとして歪み補正バッファ（Ｓkew buffer）を採用した。歪み補正バッファとは、遅延段数を任意に変更できる可変段シフトレジスタである。図２．１８に、歪み補正バッファを用いた、ＬＳＩ間パイプライン方式を示す。（ただし、図２．１８において、点線枠で示した演算部（Ａrithmetic logic）は、データユニット、メモリユニット、プロセッサユニット、および一部のリンケージユニットを含む。）

図２．１８に示すように、ＬＳＩ間のリンケージ演算を実行するＡＬＵは、それぞれ入力および出力バッファを備えている。そのため、これらのＬＳＩ間のデータ転送には、２マシンサイクルずつの時間を要する。この時間歪みは、歪み補正バッファの遅延時間を２段ずつ多くすることによって吸収できる。つまり、図２．１８に示すように、歪み補正バッファの遅延段数は、前段のＬＳＩのそれより２段ずつ大きくなっている。

図２．１８の方式は、画像データを入力する際に補正するものであるが、図２．１９に示すように、リンケージ演算を実行する直前に補正しても、ＬＳＩ間のパイプライン処理は可能である。しかし、ＩＳＰにおいては、ＰＥへのデータ供給に柔軟性をもたせ、各種の演算の実行を可能にするため、図２．１８の画像データ入力時に補正する方式を採用した。

歪み補正バッファに遅延段数は、プログラマブル制御レジスタにより、１段から１６段までの任意の段数に設定することができる。これにより、８個のＩＳＰまでの歪み補正が可能である。

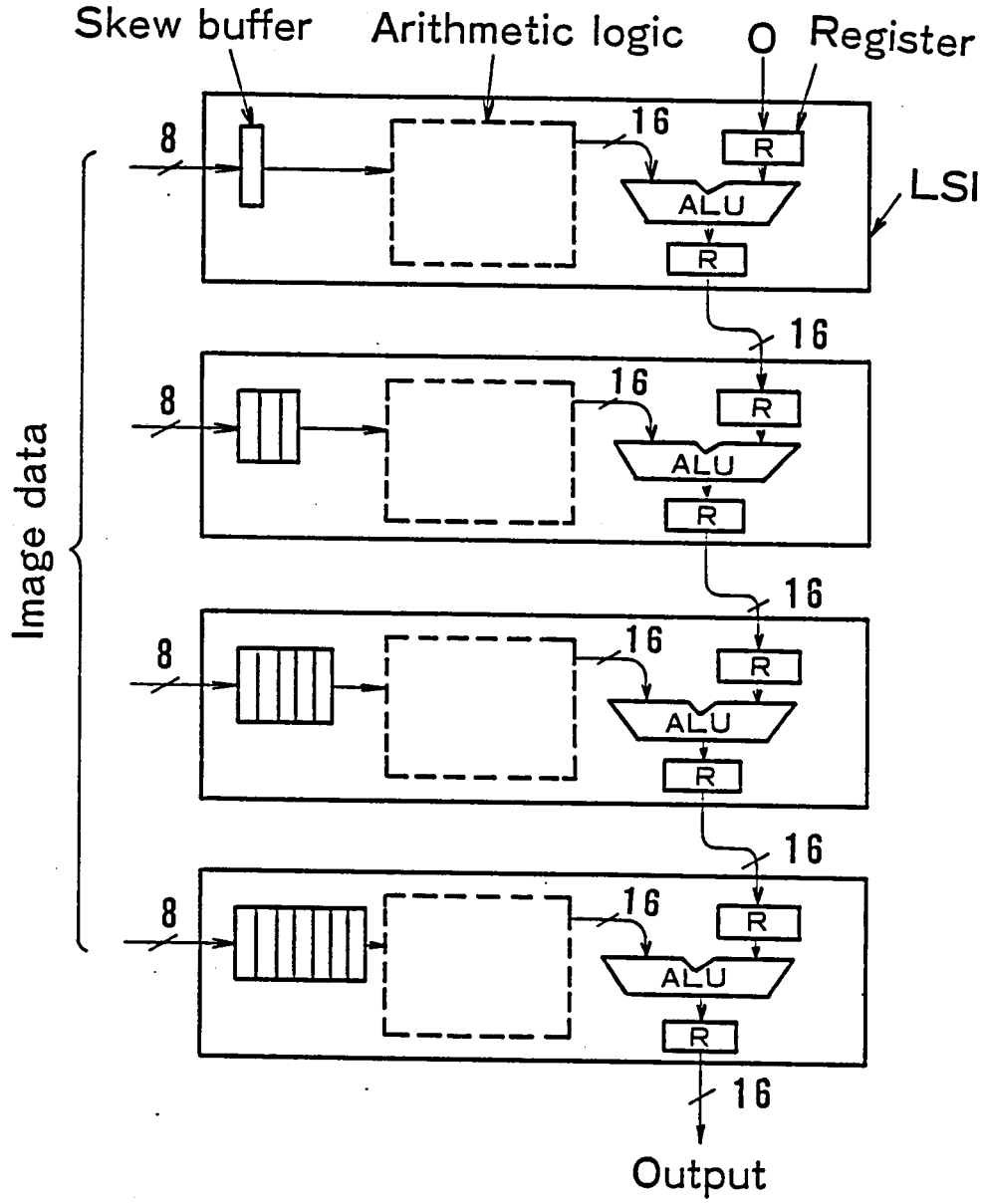


図２．１８　ＬＳＩ間パイプライン処理方式
（画像データ入力時補正）

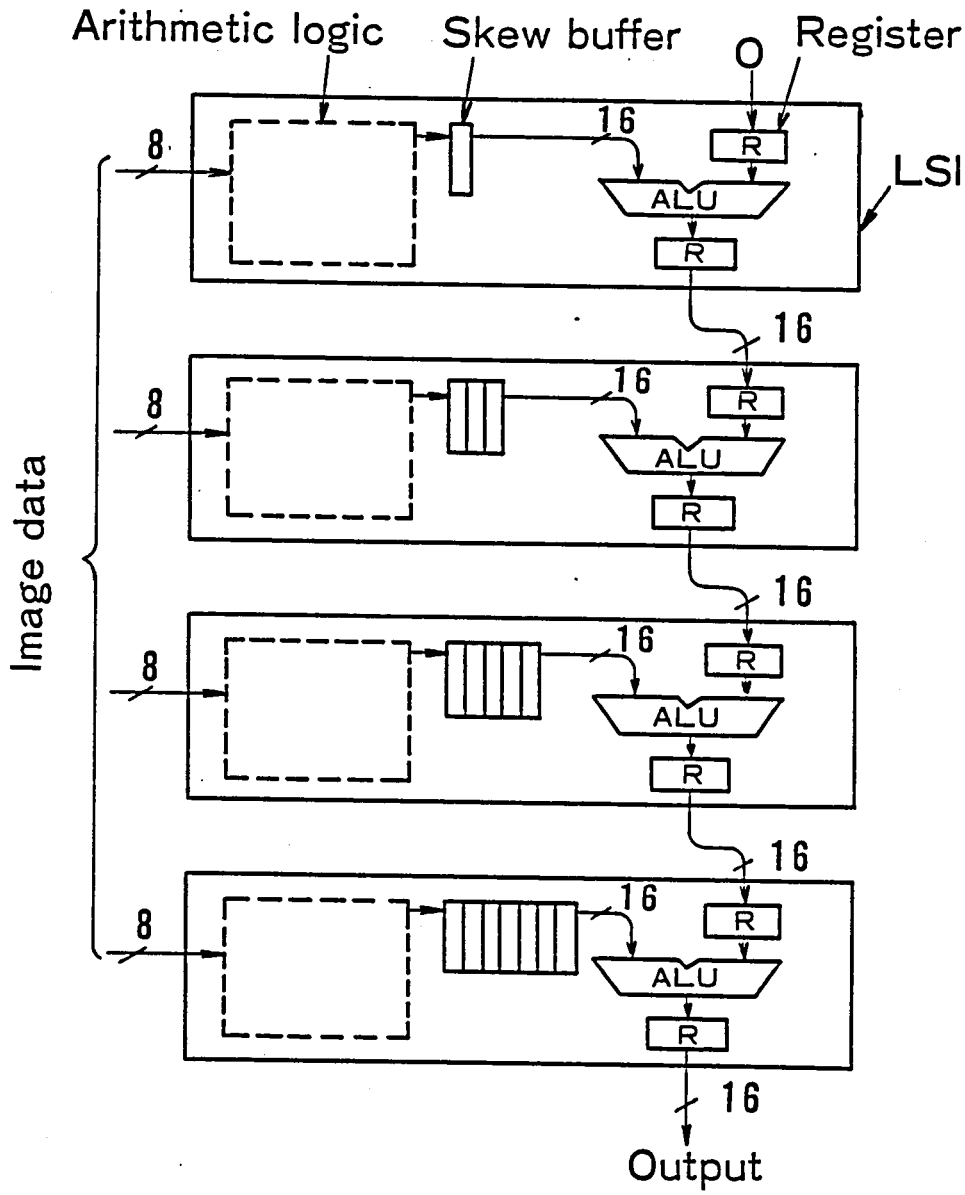


図２．１９　ＬＳＩ間パイプライン処理方式
（演算実行時補正）

２．６　ＩＳＰの基本アーキテクチャ

ここでは、前節までの検討を踏まえて決定した、ＩＳＰのアーキテクチャの概略を述べる。ＩＳＰの基本構成図を図２．２０に示す。同図に示すように、ＩＳＰは６個のユニットから構成される。以下、それぞれのユニットについて簡単に述べる。

（１）　データユニット

データユニットは、４個のＳＲ(Shift Register)などから成り、画像データの転送に寄与する。画像データバスは、入出力ともに８ビット幅で、出力バスは、カーネルの拡張に用いる。それぞれのＳＲは、次のＳＲの他、対応するＰＥに画像データを転送する。それぞれのＳＲの遅延段数は、１段から４段までの任意の段数に設定できる。これにより、ラスタ走査およびスティック走査とともに、処理の高速化にもシステムの小型化にも対応できる。また、データユニットは歪み補正バッファを持ち、複数のＬＳＩを用いる際、ＬＳＩ間のパイプライン処理を可能にしている。

（２）　メモリユニット

メモリユニットは、６４word×８bitから成るＲＡＭで構成され、空間積和演算における荷重係数や、パターンマッチングにおけるテンプレートなどのマスクデータを記憶する。マスクデータは、演算実行に先立って書き込まれる。演算実行時は、端子からのアドレス情報に従って、４バイトのマスクデータが読み出され、それぞれのＰＥに１バイトずつデータを供給する。

（３）　プロセッサユニット

プロセッサユニットは、４個のＰＥから構成され、それぞれのＰＥは、加算器(ＡＬＵ－Arithmetic Logic Unit)と乗算器(multiplier)を一つずつ含んでいて、データユニットとメモリユニットから、それぞれ８ビットのデータを入力し、リンケージユニットに、１６ビットの演算結果を出力する。それぞれのＰＥは、コントロールユニットからの制御信号に従って、ＳＩＭＤ(Single Instruction Multiple Data stream)の形態で並列動作する。ＰＥ内の演算器の機能については、第３章で詳しく論ずる。

（４）　リンケージユニット

リンケージユニットは、４入力ＡＬＵと２入力ＡＬＵから構成されている。４入力ＡＬＵはＰＥ間の演算に、２入力ＡＬＵはＬＳＩ間のリンケージ演算に寄与する。ＰＥ節約方式によりカーネルを拡張する場合、２入力ＡＬＵはアキューミュレータとしても動作する。リンケージユニットのデータバスは、入出力ともに１６ビット幅である。本ユニット内の演算器の機能については、第３章で詳しく論ずる。

（５）　エバリュエーションユニット

エバリュエーションユニットは、２個の比較器などから構成され、２値化処理やクラスタリング処理などに寄与する。本ユニットの構成については、第４章で考察する。

（６）　コントロールユニット

コントロールユニットは、プログラマブル制御レジスタなどから構成され、データフローやそれぞれのユニットの機能を制御する。プログラマブル制御レジスタの内容を変更することにより、種々の画像演算を実行できる。

なお、ＩＳＰのチップ写真と基本仕様を、それぞれ図２．２１と表２．１に示す。

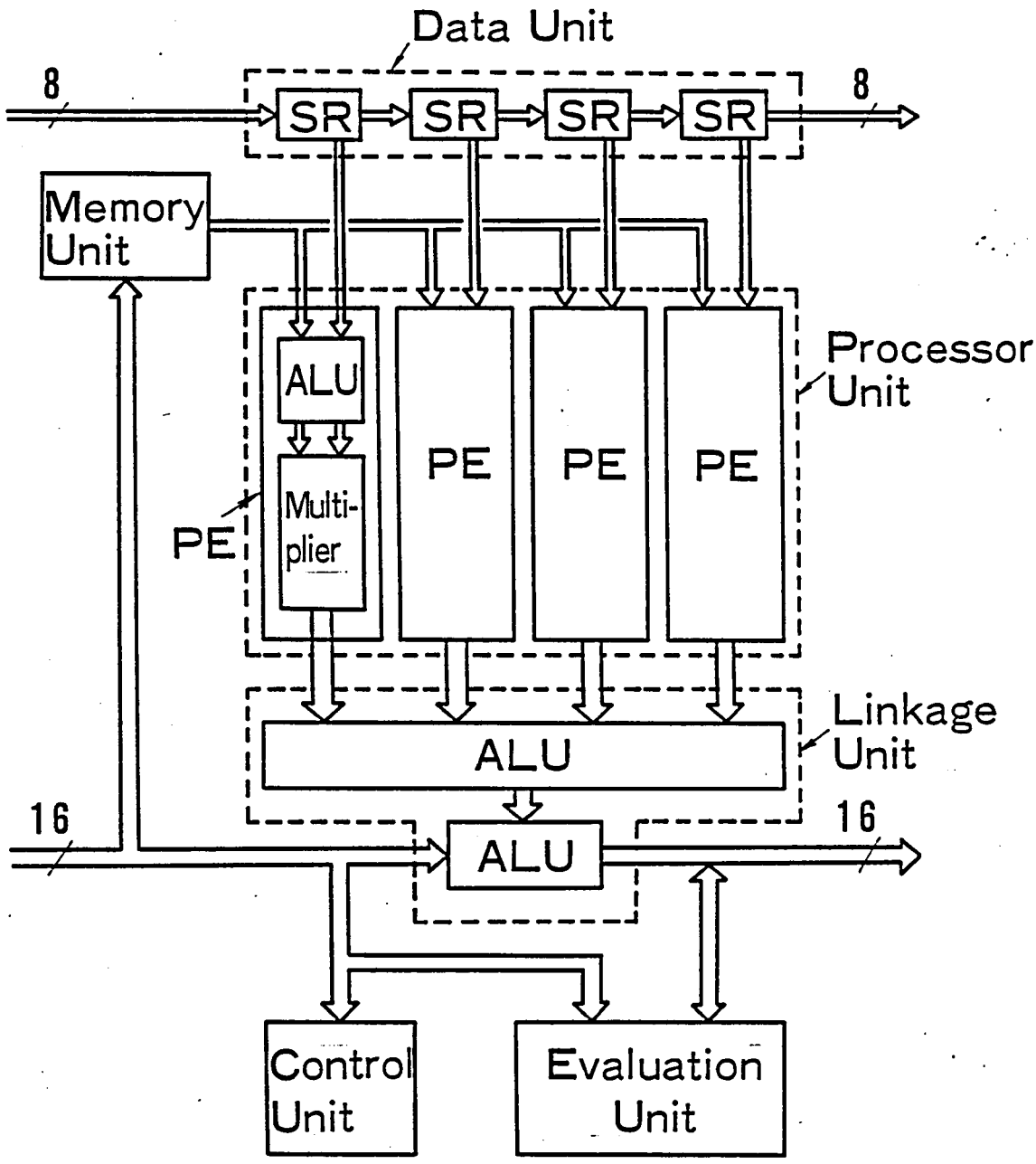


図２．２０　ＩＳＰの基本構成図

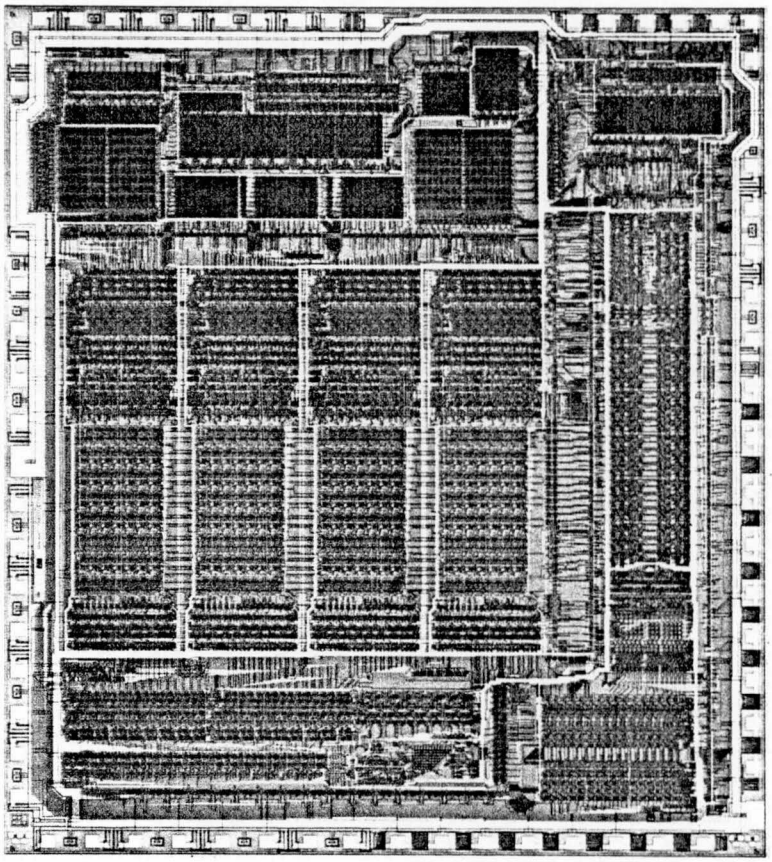


図２．２１　ＩＳＰのチップ写真

表２．１　ＩＳＰの基本仕様

テクノロジー	３μｍ　ＣＭＯＳ
トランジスター数	～６１，０００
チップサイズ	７．７２×８．６４ｍｍ
電源電圧	５Ｖ
マシンサイクル	１６７ｎｓ
消費電力	～４００ｍＷ
パッケージ	６４ｐｉｎ　ＤＩＰ

２．７　まとめ

濃淡画像処理技術を一般産業へ応用することを目的として、画像処理用ＬＳＩ－ＩＳＰを開発した。ＩＳＰは、一つの出力画素を算出するのに用いる入力画像と同数のＰＥを用意して、局所的に並列演算する局所並列型のＬＳＩであり、ここでは、ＩＳＰの基本アーキテクチャを明らかにした。

ＩＳＰの基本アーキテクチャを検討する上で、濃淡画像における最も基本的な演算である、空間積和演算を題材とした。その結果、次の７点を基本構想として、ＩＳＰのアーキテクチャを決定した。

（１）　１チップに４個のＰＥを搭載する。

（２）　４個のＰＥは１×４の構成とする。

（３）　画像データは遅延回路を介して、荷重係数は内蔵ＲＡＭから、それぞれＰＥに供給される。

（４）　カーネルは、ＬＳＩの追加もしくは時分割使用、いずれによっても拡張できる。

（５）　ＬＳＩ間のリンケージには、専用の入出力端子と演算器を用意する。

（６）　ＬＳＩ内およびＬＳＩ間に、マシンサイクルが１６７ｎｓのパイプライン処理を構築する。

（７）　プログラマブル制御レジスタによる構成制御を用いる。

ＩＳＰは、ＰＥの数を増やすＰＥ増設方式と、ＰＥを時分割に用いるＰＥ節約方式の、二つの方法によりカーネルを拡張できる。ＰＥ増設方式によりカーネルを拡張する場合、入力画像はテレビ画像と同じラスタ走査により走査されればよい。ＰＥ節約方式によりカーネルを拡張する場合は、新たに提案したスティック走査により走査されなければならない。

ＰＥ増設方式とラスタ走査によると、テレビ画像を実時間処理できる、高速画像処理システムが構築できる。一方、ＰＥ節約方式とスティック走査を用いると、処理速度とのトレードオフで、非常にコンパクトな画像処理システムを構築できる。

２．８　第２章の参考文献

[1]Fukushima, T., Kobayashi, Y., Hirasawa, K., Bandoh, T., Ejiri, M. and Kuwahara, H. : An Image Signal Prosessor, IEEE ISSCC Digest of Technical Papers, Vol.26, pp.258～259 (1983).

[2]福島　忠、小林芳樹、平沢宏太郎、坂東忠秋、江尻正員、柏岡誠治、桑原　洋：画像処理用ＬＳＩ－ＩＳＰの開発、情報処理学会第２６回全国大会講演論文集、pp.939～940 (1983).

[3]Tsoras, J. : The Massively Parallel Processor (MPP) Innovation in High Speed Processor, AIAA Computers in Aerospace Ⅲ Conference, pp.196～201 (1981).

[4]Sudo, T., Nakashima, T., Aoki, M. and Kondoh, H.: An LSI Adaptive Array Processor, IEEE ISSCC Digest of Technical Papers, Vol. 25, pp.122～123 (1982).

[5]Lougheed, R. M. and McCubbrey, D. L. : The Cytocomputer - A Practical Pipelined Image Processor, IEEE 7th Annual International Symposium on Computer Architecture (1980).

[6]Nudd, G. R. : Image Understanding Architecture, National Computer Conference, pp.377～390 (1980).

[7]福島　忠、小林芳樹、平沢宏太郎、坂東忠秋、江尻正員：画像処理用ＬＳＩ－Image Signal Processor のアーキテクチャ、電子通信学会論文誌(C)、Vol. J66-C, No. 12, pp.959～966 (1983).

第３章　画像処理用ＬＳＩ－ＩＳＰの多機能化

３．１　まえがき

画像の最小構成要素である画素のそれぞれに、２ビット以上の情報を与えた濃淡画像や色彩画像を処理でき、かつ、一般産業に幅広く活用されうるＬＳＩを開発する際、そのＬＳＩが、①ビデオレート（１６７ｎｓ／画素）で高速処理できる、②任意の大きさのカーネルを扱える、③種々の基本演算を実行できる、以上三つの特徴を備えることを目標とした[１]。①の高速処理は２５６×２５６画素サイズ程度の画像を、テレビカメラの走査速度に追随した、実時間での処理を可能とする。②の拡張性は、応用によってサイズの異なるカーネル（局所演算領域）への対応性を増加させる。また、③の多機能性は、ＬＳＩの適応可能な応用分野そのものを拡げるために、欠かすことができない。

これら三つの開発目標のうち、①の高速処理と②の拡張性を中心として、第２章において、画像処理用ＬＳＩ－ＩＳＰ（Ｉmage Ｓignal Ｐrocessor）の基本アーキテクチャを明らかにした。基本アーキテクチャは、最も基本的な局所近傍処理の一つである空間積和演算を題材として検討した。ＩＳＰは、ＰＥ（Ｐrocessor Ｅlement）４個を内蔵した並列処理方式とするとともに、パイプライン処理方式も合わせて採用し、ビデオレートの実時間処理を達成している。また、カーネルの拡張に対しては、ＩＳＰをビルディングブロック的に追加して、並列に動作するＰＥ数を増やすＰＥ増設方式と、ＰＥを時分割に使用して、少ないＬＳＩ数でカーネルを拡張するＰＥ節約方式とを実現した[２]。

③の多機能性は、画像処理を一般産業へ適用する上から、高速処理・拡張性と並んで、最も重要な要素の一つである。なぜなら、さまざまな応用に受け入れられるような、普遍的なアプローチが画像処理に確立しておらず、適用する対象毎に異なる機能が要求されるため、基本的な画像演算の多くを実行できる多機能性を備えることが、ＬＳＩの汎用性を高める上で欠かせないからである。

しかしながら、ハードウェア回路には、本来、柔軟性がないものである。特に、パイプライン処理方式を採用した演算回路には、多種の実行機能を期待することはできない。なぜなら、パイプライン処理方式は、特定の演算を高速に処理するため、データの流れを固定化（パイプ化）して、いくつものステージに分割しており、柔軟性を犠牲にしているからである。

そこで、ＩＳＰの開発においては、パイプライン処理方式の持つ柔軟性のなさという短所を、次の二つの考え方で補うことを考えた。一つは、いく段ものパイプラインステージから成る演算部に入力されるデータに、多種類の組合せを可能とすることである。もう一つは、演算部の個々のパイプラインステージに、オペレーションレベルでの機能をいくつか持たせ、任意の選択、組合せができるようにして、演算部全体として、多彩な演算機能を実現しようとするものである。

この二つの考え方を基として、ここでは、第２章で明らかにした基本アーキテクチャの上で、２値・濃淡・色彩画像の種々の基本演算を実現する方法についての考察を加え、演算データの制御と回路構成および機能について、検討した内容を論ずる[３]。

３．２　多機能化の考察

３．２．１　多機能化へのアプローチ

前章で明らかにした画像処理用ＬＳＩ－ＩＳＰの基本アーキテクチャを基に、２値・濃淡・色彩画像に対する種々の基本的な局所近傍演算を実現する方法として、二つのアプローチを採った。一つは、画像処理ＬＳＩ内の構成要素の有機的な結合に着目した構造的アプローチであり、もう一つは、それぞれの構成要素に持たせる機能内容に着眼した機能的アプローチである。

多機能化を検討する上で、前章で明らかにした図２．２０の基本アーキテクチャを横断的に考察すると、ＬＳＩの演算実行部は以下のように見ることができる。

（１）　プロセッサユニットのＰＥは、メモリユニットからのマスクデータにより、画素データを変換する機能を担う。

（２）　リンケージユニット内の４入力ＡＬＵは、４個のＰＥの出力データを統合する機能を担う。

（３）　リンケージユニット内の２入力ＡＬＵは、複数のＬＳＩに跨って、同ユニット内の４入力ＡＬＵの出力データを再統合する機能を担う。

以下、（１）を画素データ変換機能、（２）を変換データ統合機能、（３）を統合データ再統合機能と呼ぶことにする。

多機能化を検討する手順としては、先ず上記の三つの構成的機能を念頭に置いた構造的アプローチを採り、ＬＳＩの内部構成を明確にした。そして、その内部構成を前提として、機能的アプローチを採ることにした。

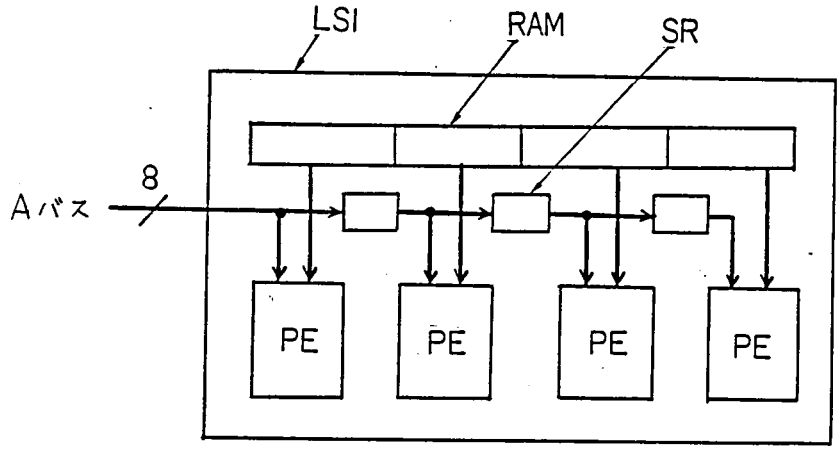


図３．１　ＰＥへのデータ供給方式（１）

３．２．２　構造的観点からの検討

図２．２０に示す基本アーキテクチャは、ビデオレート（１６７ｎｓ／画素）で高速処理するため、並列処理に加えてパイプライン処理を採用している。また、ＬＳＩ化における論理設計・レイアウト設計・論理検証などの作業工数を低減させるため、内部構成は規則性を重視した構成となっている[４]。そのため、種々の画像演算を実行させるべき手段は、これらの点を考慮したものでなければならない。そこで、並列処理・パイプライン処理と、内部構成の規則性を前提として、プロセッサユニットとリンケージユニットのそれぞれの演算要素に、どのようにデータを供給するか、という観点から構成を検討することにした。

図２．２０の基本アーキテクチャにおいて、プロセッサユニットの４個のＰＥには、データユニットとメモリユニットから、それぞれ１バイトずつデータが供給される。データユニットからのデータは画素データであり、メモリユニットからのデータは演算オペランドである。これらＰＥへのデータ供給は、図３．１のように図示できる。図３．１から分かるように、ＳＲ（Shift Register）により順次遅延されてＰＥに供給される画素データは、ＲＡＭ（Random Access Memory）から与えられる演算オペランドによって変換されることになる。

この基本アーキテクチャは、前章で述べたように、空間積和演算の実行に焦点を当てて検討されたものである。しかし、空間積和演算以外にも有効な画像演算が、これまで数多く報告されている。その中には、表３．１に示す各種の１次微分オペレータがある[５～12]。

表３．１　各種の１時微分オペレータ

(1)	$\sqrt{(a-d)^2+(b-c)^2}$ (Roberts オペレータ)
(2)	$\lvert a-d\rvert+\lvert b-c\rvert$
(3)	$[\lvert a-b+c-d\rvert+\lvert a+b-c-d\rvert]/2$
(4)	$\sqrt{(A+B+C-G-H-I)^2+(A+D+G-C-F-I)^2}$
(5)	$\lvert A+B+C-G-H-I\rvert+\lvert A+D+G-C-F-I\rvert$
(6)	$\lvert E-A\rvert+\lvert E-C\rvert+\lvert E-G\rvert+\lvert E-I\rvert$
(7)	$\lvert\max(A,B,C,D,F,G,H,I)-E\rvert$
(8)	$\mathrm{sign}(B-H)\cdot\lvert\min(A,B,C)-\max(G,H,I)\rvert$
(9)	$\lvert A+2B+C-G-2H-I\rvert+\lvert A+2D+G-C-2F-I\rvert$ (Sobel オペレータ)

a	b
c	d

A	B	C
D	E	F
G	H	I

表３．１に示す各種の１次微分オペレータは、空間積和演算と他の演算に分解して実行できる。たとえば、表３．１(９)のＳobelオペレータは、図３．２に示す２個のマスクデータを用いれば、２回の３×３空間積和演算と、２回の絶対値演算、および１回の加算演算で実行することができる。しかし、このようにオペレーションを分解すると、これまで検討した基本アーキテクチャの中では、１度のパイプライン処理で実行できない。この方法では、表３．１のどの１次微分オペレータでも、空間積和演算を２回以上実行しなければならなくなるので、少なくとも２倍以上の処理時間を必要とすることになる。このために、空間積和演算を実行する回路を２組以上用意しなければ、高速処理は望めない。これは、ハードウェア設計上大きなディメリットである。

そこで、空間積和演算のように、マスクデータによって画素データを変換するのではなく、近傍の画素データによって、注目の画素データを変換することを検討した。ここでは、メモリユニットのＲＡＭからＰＥへ演算オペランドを供給する代りに、注目の画素データに加えて、近傍の画素データをＰＥに供給することを考えた。つまり、それぞれのＰＥには、２バイトずつ画素データが供給されることになる。

1	2	1
0	0	0
−1	−2	−1

1	0	−1
2	0	−2
1	0	−1

図３．２　Ｓobelオペレータを実行するためのマスクデータ

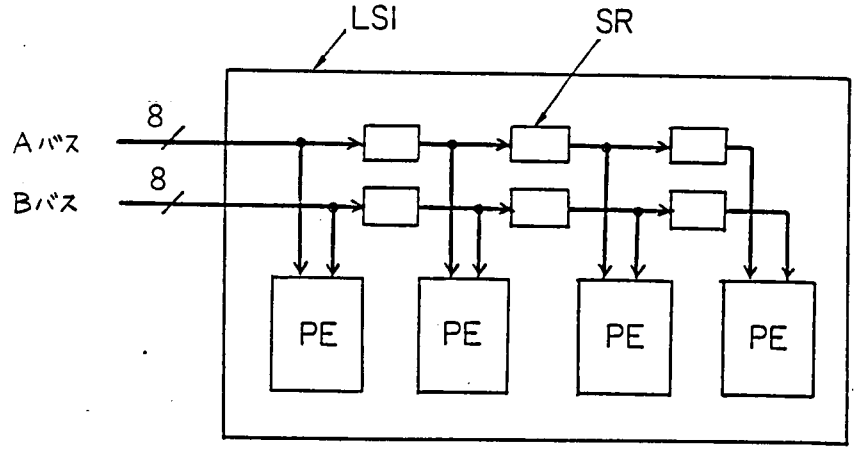


図３．３　ＰＥへのデータ供給方式（２）

ＰＥへの画素データの供給は、表３．１に示す各種の１次微分オペレータを効率よく実行できることを目指して、二つの方式を検討した。一つは図３．３に示すように、二つの８ビット画素データを、ともにＳＲにより順次遅延させて、それぞれのＰＥへ供給する方式である。もう一つは図３．４に示すように、ＰＥに供給される一方の画素データは、ＳＲにより順次遅延させられるが、もう一方の画素データを、４個のＰＥに同時に供給する方式である。前者によれば、表３．１の(１)や(２)のオペレータのように、上下に隣接する二つの画素列間の演算をＰＥで実行できるようになり、後者によれば、表３．１の(６)や(７)のように、３×３のカーネルの中央画素Ｅを用いる演算を、それぞれのＰＥで並列に実行できるようになる。

画像処理用ＬＳＩ－ＩＳＰは、図３．１に示すデータ供給方式を基本としているが、図３．３および図３．４のデータ供給方式を実現する上で問題となるのは、新たなデータバスを設けることによって端子数が増加することと、配線面積が増加することである。そこで、図３．３と図３．４のデータ供給方式を採用する上で、端子数や配線面積を増加させない方法を検討した。つまり、図３．１のＡバスと同様、図２．２０におけるデータユニットへの入力バスを、図３．３、図３．４におけるＡバスに用いる一方、図２．２０におけるデータユニットからＬＳＩ外部への出力バスを、双方向性として、図３．３、図３．４のＢバスに転用することにした。この結果、双方向バスの入出力切り換え制御用に、端子数を１本増加するだけで対応できた。

一方、空間積和演算と同様に、演算オペランドを用いて画素データを変換する画像演算でも、拡大・縮小・回転などの移動を行なった場合などに実施される補間演算や、赤・緑・青の光の三原色毎の濃淡値を用いた色彩処理などに対しては、別の角度からの検討が必要である。それは、画素データの同時供給という観点である。空間積和演算や１次微分オペレータの場合、処理対象となるカーネルは、画像走査にともなって移動してゆく。そのた

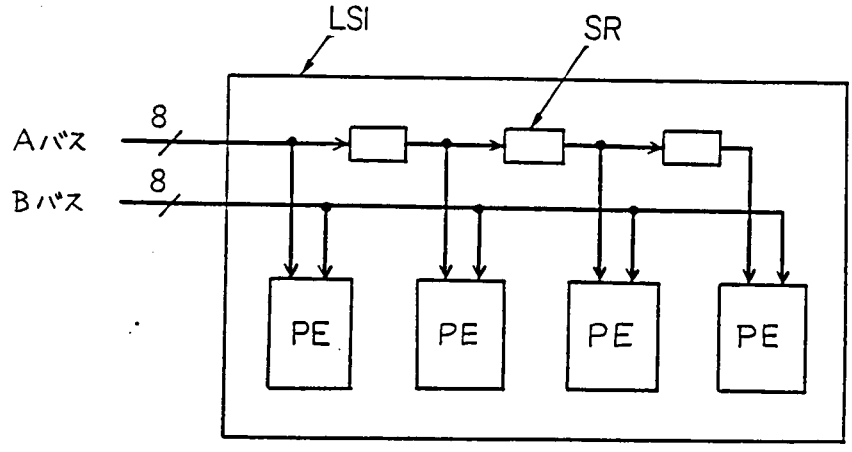


図３．４　ＰＥへのデータ供給方式（３）

め、図３．１などのようにＳＲにより画素データを順次遅延してＰＥに供給するのが有効であった。しかし、補間演算では、演算に用いられる画素データは、基本的には任意であり、ＳＲにより遅延して、ＰＥに規則的に供給することはできないため、逐次的に入力しなければならなくなる。また色彩処理においては、三原色に分解したを３枚の濃淡画像を別々に処理するのでなく、三つの色の濃度値に依存するような処理（たとえば、色彩空間における二つの画素の距離を計算する処理）を行う場合なども、図３．１などのような入力方式では、やはり画素データを逐次的に入力する必要がある。

画素データを逐次的に入力することは、入力サイクルを演算サイクルと切り離して高速化しない限り、データ入力がボトルネックとなって処理速度を低下させることになる。画像処理用ＬＳＩ－ＩＳＰの場合、処理速度の低下を防ぐには、４個のＰＥへ画素データを供給するために、データ入力および転送速度を、演算速度の４倍にしなければならない。演算サイクルが１６７ｎｓであることと、用いるプロセス技術が３μｍＣＭＯＳ（Complementary Metal Oxide Semiconductor）であることを考えると、これは、ほとんど不可能である。この不都合を解消するため、画素データを並列にＰＥへ供給することを検討した。

複数の画素データを、並列にＰＥへ供給する場合に問題となるのは、やはり、入力端子数の増加と配線面積の増加である。４個のＰＥに８ビットずつ画素データを供給するとなると、入力端子数は合計３２本必要になる。この内１６本は、図３．３などのように、図２．２０におけるデータユニットへの入出力バス、ＡおよびＢバスをそのまま利用できる。また残りの１６本については、図２．２０におけるリンケージユニットへの入力バス、ＬＩバスを用いることにした。これにより、新たに端子を増設する必要は生じない上、既設のデータバスの一部を共用でき、配線面積の増加を必要最小限に抑えることができる。ただし、２個以上のＬＳＩを使用する場合、ＬＳＩ間のリンケージ処理を実行することはできない。

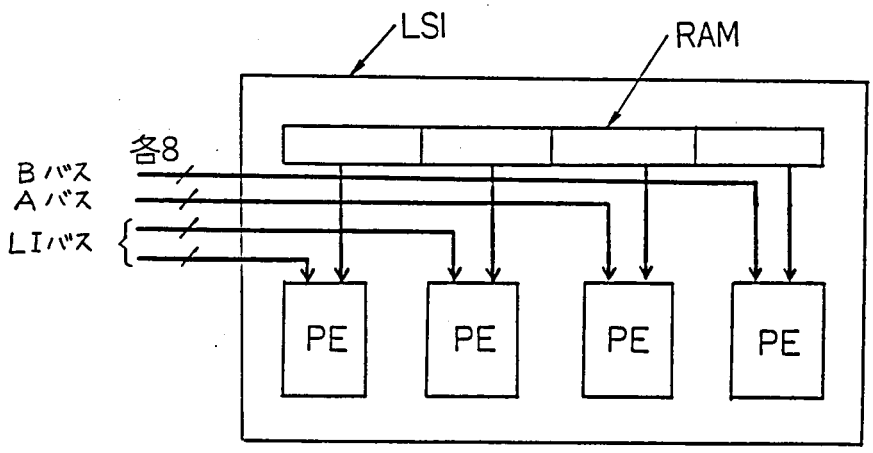


図３．５　ＰＥへのデータ供給方式（４）

これらの検討の結果、４個のＰＥに、それぞれ１バイトの画素データを別々に供給できるよう、図３．５に示すデータ供給方式も合わせて採用することにした。図３．５におけるＡ、Ｂ、ＬＩバスは、それぞれ図２．２０における、データユニットへの入力バス、データユニットから外部への出力バス、リンケージユニットへの入力バスに対応させた。

ここで一つ考慮に入れなければならないのが、稼動するＰＥの選択についてである。なぜなら、空間積和演算、１次微分演算、補間演算、色彩演算のいずれにおいても、ＬＳＩ内の４個のＰＥをすべて使用するとは限らないからである。たとえば、３個のＩＳＰを用いて３×３のカーネルを処理する場合、ＩＳＰ内で用いられるＰＥは、それぞれ３個だけである。そこで、使用するＰＥを外部から任意に選択する方法を検討した。基本的には、端子数を増加させないことを重視して、演算回路の機能指定と同様に、ＬＳＩに内蔵するプログラマブル制御レジスタにより、稼動するＰＥを選択できる方式を採用した。制御レジスタ内に、ＰＥそれぞれに対して２ビットの制御ビットを確保し、ＰＥの稼動および停止を独立して指定できるようにした。（停止時のＰＥ出力値は、０、$2^{15}-1$、もしくは-2^{15}から選択できる。）つまり、それぞれのＰＥは、

① 所定の演算を実行し、結果を出力する。
② ０の値を出力する。
③ $2^{15}-1$の値を出力する。
④ -2^{15}の値を出力する。

のいずれかを、他のＰＥと独立に実行する。これにより、任意の形状のカーネルによる近傍演算を実行できることになる。

以上論じた内容は、濃淡画像、もしくは３枚の濃淡画像から成ると見れる色彩画像の処理に適合する。もちろん、１画素が１ビットから成る２値画像に対しても、ビット幅の小さな濃淡画像として処理することも可能である。しかし、８画素を１単位として、濃淡画

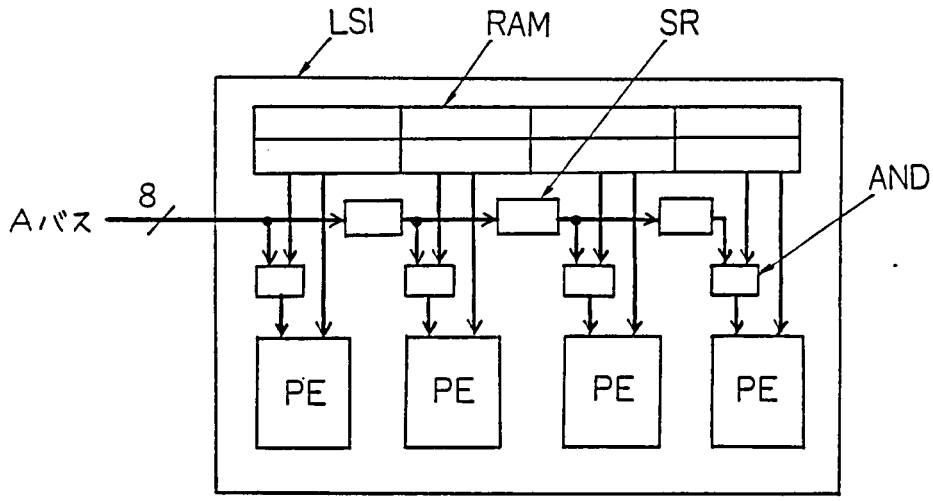


図３．６　ＰＥへのデータ供給方式（５）

像の一つの画素に対応させて処理する方が効率がよい。この方法では、１個のＬＳＩで、８ビット×４ＰＥ、合計３２画素の２値データを並列に処理できる。しかしこの場合、上に述べた稼動ＰＥの選択だけでは、任意の形状のカーネルを形成することはできないという問題がある。

任意形状のカーネルを形成するためには、ビット単位でのカーネル指定手段が必要である。ビット単位の指定となると、そのための情報としてそれぞれのＰＥへ、新たにもう１バイトのデータを供給しなければならない。このカーネル指定用データの供給方法には、ＬＳＩ外部から供給する方法と、あらかじめ内部に記憶しておく方法とが考えられるが、端子数を増加させないことを優先させると、後者の方法を採らざるを得ない。カーネル指定用データをＬＳＩ内部に記憶する方法には、専用のレジスタを用意する方法と、荷重係数用のＲＡＭを兼用する方法とがある。しかし、前章で述べたＰＥ節約方式によるカーネル拡張にも対応できる上、ハードウェアの増加を抑える上で効果のある、ＲＡＭからのデータ供給方式を採用した。

つまり、２値画像処理におけるカーネルを任意の形状に設定できるよう、図３．６に示すように、メモリユニットのＲＡＭからそれぞれのＰＥに、２バイトのデータを供給することにした。１バイトはカーネル指定用、もう１バイトを演算オペランド用である。図３．６において、演算オペランド用のデータは、ＲＡＭから直接ＰＥに供給されるが、カーネル指定用のデータは、画素データとビット毎に論理積がとられた後、ＰＥへ入力される。[13]。

３．２．３　機能的観点からの検討

機能的な観点からは、前項で検討した構成におけるデータフローを基にして、種々の画像演算の実行に必要とされる演算機能を検討することにする。ここでは、局所近傍演算として分類される代表的な演算として、空間積和演算や表３．１に示した各種の１次微分オペレータのほか、表３．２に定義される各種の濃淡および色彩画像演算、さらに、２値画像演算としてパターンマッチング機能を取りあげる[14]。そして、これらの画像演算を実現する上から、プロセッサユニットのＰＥの担う画素データ変換機能、リンケージユニットの４入力ＡＬＵの担う変換データ統合機能、リンケージユニットの２入力ＡＬＵの担う統合データ再統合機能、に要求される演算オペレーションについて論ずる。

表３．２　各種の濃淡および色彩画像演算

空間積和演算	$g(x,y)=\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}\cdot f(x+i-1,y+j-1)$
残差検定法	$g(x,y)=\sum_{i=1}^{n}\sum_{j=1}^{m} \lvert t_{ij}-f(x+i-1,y+j-1)\rvert$
４点線形補間	$g(x,y)=(1-\Delta x)(1-\Delta y)\cdot f(i,j)$ $+\Delta x(1-\Delta y)\cdot f(i+1,j)$ $+(1-\Delta x)\Delta y\cdot f(i,j+1)$ $+\Delta x\Delta y\cdot f(i+1,j+1)$
色彩濃度変換	$g(x,y)=\alpha R(x,y)+\beta G(x,y)+\gamma B(x,y)$
色彩距離演算	$g(x,y)=\sqrt{\{\alpha-R(x,y)\}^2+\{\beta-G(x,y)\}^2+\{\gamma-B(x,y)\}^2}$

f：濃淡入力画像
g：濃淡出力画像
w,α,β,γ：荷重係数
t：テンプレートデータ
R：色彩画像 Red 成分濃度
G：〃 Green 〃
B：〃 Blue 〃
Δx：隣接画素 $f(i,j)$ との x 軸方向の差異
Δy：〃 〃 との y 軸方向の差異

表３．３は、上記の各種画像演算の実行につき、要求される演算オペレーションを、プロセッサユニットとリンケージユニットの担う三つの機能に分割した結果を示す。表３．３からは次の五つの結論が得られる。

（１） 画素データ変換機能、変換データ統合機能、統合データ再統合機能、のいずれの機能も、連続する二つの演算部で実現できる。

（２） 画素データ変換機能に対しては、加減算を主なオペレーショとする演算部と、乗を主なオペレーションとする演算部が必要である。

（３） 変換データ統合機能と統合データ再統合機能に対しては、加減算を主なオペレーションとする演算部と、絶対値算出オペレーションを備えた演算部が必要である。

（４） 統合データ再統合機能には、ＳＱＲＴ(square root)機能も要求される。

（５） いずれの演算部にも、入力データをそのまま出力するＮＯＰ(no operation)機能が要求される。

これらの結果を踏まえ、さらに他の画像演算の実行を考慮して、三つの機能を担う、プロセッサユニットのＰＥやリンケージユニットのＡＬＵの、演算回路構成を決定することにした。

表３．３　各機能に要求される演算オペレーション

画像演算	画素データ変換機能	変換データ統合機能	統合データ再統合機能
一次微分オペレータ (1)	−, ＊	NOP	+, $\sqrt{\ }$
〃 (2)	−, ||	NOP	+
〃 (3)	+／−	+／−, ||	+, 1/2
〃 (6)	−, ||	+	+
〃 (7)	−, ||	max	max
〃 (8)	NOP	min/max	−, ||
空間積和演算	＊	+	+
残差検定法	−, ||	+	+
4点線形補間	＊	+	+
色彩濃淡変換	＊	+	+
色彩距離演算	−, ＊	+	+, $\sqrt{\ }$
パターンマッチング	ENOR, CNT	+	+

＋：加算　　max：最大値抽出　　NOP ：no operation

−：減算　　min：最小値抽出　　ENOR：排他的論理和の否定

＊：乗算　　\|\| ：絶対値抽出　　CNT ：ビット"1"の総数計算

$\sqrt{\ }$：平方根　　1/2：1/2を乗ずる　　+/- ：加算もしくは減算

３．３　多機能性を実現するアーキテクチャ

３．３．１　演算データの構成制御

第３．２．２項の検討を踏まえて決定した、画像データと演算オペランドの制御構成を図３．７に示す。同図は、データユニット、メモリユニットのＲＡＭ、および４個のＰＥの一部などを含んでいる。図３．７には二つの歪み補正バッファがあるが、Ａバスのものは、ＬＳＩ間のパイプライン処理を可能にするものであり、Ｂバスのものは、Ａバスとの時間歪みを補正するものである。Ｂバスの歪み補正バッファは、データユニットのデータ制御と合わせて、表３．１に示した各種の１次微分オペレータの実行を可能にしている。

メモリユニットのＲＡＭは、２値画像処理におけるカーネルの任意性を保つため、ＰＥ１個に対して２バイト、合計８バイトのデータを並列に読み出させる構成でなければならない。しかし、並列に読み出す８バイトの半分の４バイトは、２値画像処理におけるカーネル指定以外の用途に用いられることがないため、通常の並列読み出し構成とすると、ＲＡＭ容量の半分は、ほとんどの処理で用いられないものとなり、面積の使用効率が悪くなる。チップサイズの小型化からは、ＲＡＭの容量を減らしたい一方、前章で述べたＰＥ節約方式によるカーネル拡張を考えると、大きな容量のＲＡＭが望ましいことは言うまでもないことである。この相反する要求を満たすため、すべてのデータを、４バイト単位で個々に読み出せるだけでなく、一つのアドレスで、４バイト単位のデータを２組読み出すことも可能なＲＡＭの構成とした。

メモリユニットのＲＡＭは、図３．８に示すように、四つのブロックＭＢ（Memory Block）から成っていて、プロセッサユニットの４個のＰＥに、それぞれ対応づけられている。それぞれのＭＢは、チップサイズの点から１６word×８bitの構成とし、アドレス信号はすべてのＭＢに共通とした。読み出しアドレスは、ＬＳＩ外部から４ビットで与えられ、それぞれのＭＢは、１アドレス１ワード、もしくは１アドレス２ワードの読み出しを行う。１アドレス２ワードの場合、読み出しアドレスをｘ番地とすると、それぞれのＭＢのｘ番地の内容が、演算オペランドとして読み出され、また同時に、[ｘ＋８]mod16 番地の内容が、２値画像処理用のカーネル指定データとして、対応するＰＥへ読み出される。つまり、１アドレス２ワード読み出しにより、２値画像のパターンマッチング処理において、ＲＡＭ容量の半分を荷重係数用として、残りの半分をカーネル指定用として使用できることになる。

なお、メモリユニットのＲＡＭへのデータ書き込みは、端子数の増加を避けるため、リンケージユニットへの入力バスである１６ビットのＬＩバスを用いている。この時のＬＩバスのビット構成を図３．９に示す。同図に示すように、ビット０－７は書き込みデータであり、ビット８－１１は書き込みＭＢを指示し、ビット１２－１５は書き込みアドレスである。つまり、ビット８－１１のうちの複数ビットを１にすることにより、複数のＭＢの同一アドレスへ、同じデータを同時に書き込むことができる[15]。

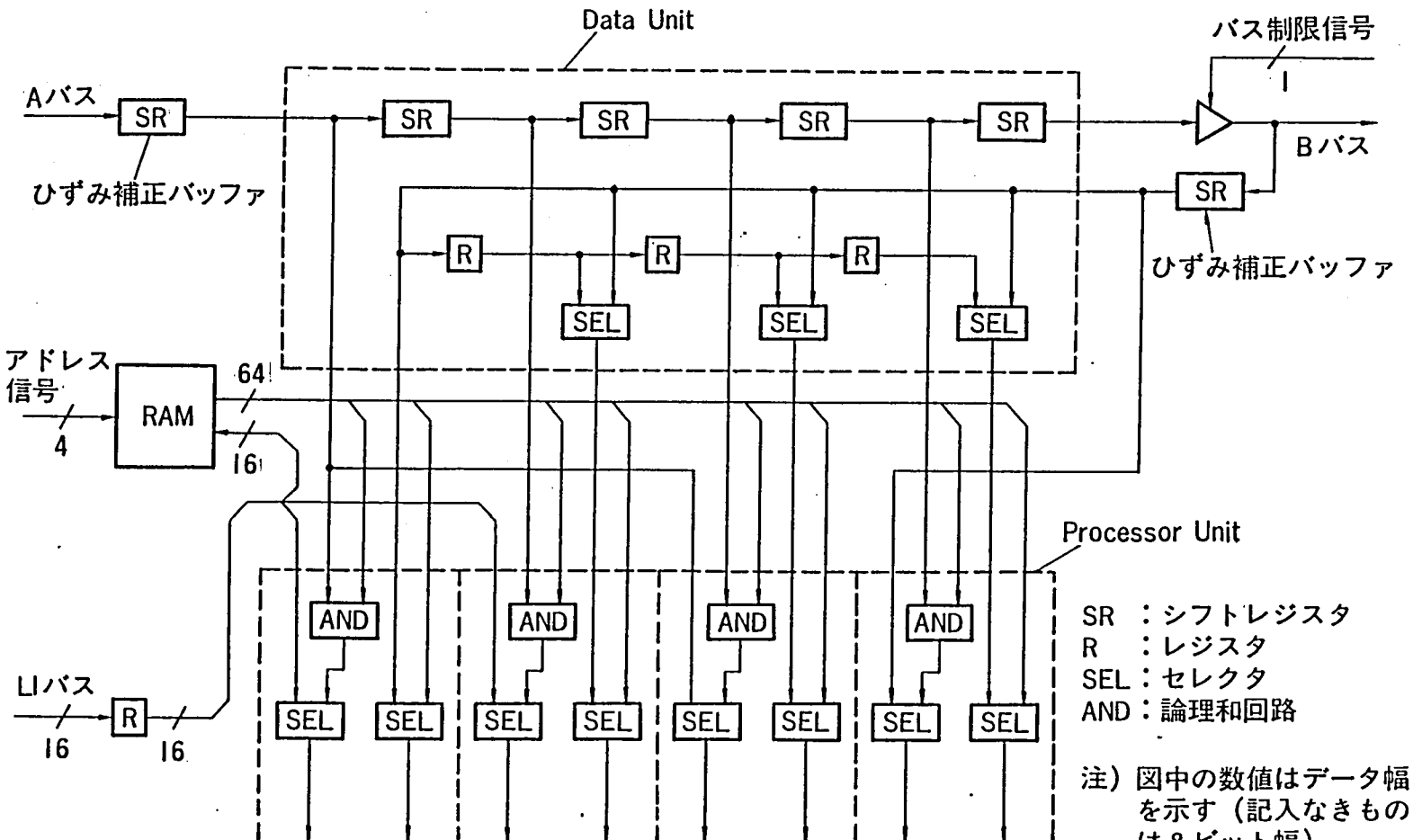


図３．７　画像データと演算オペランドの制御構成

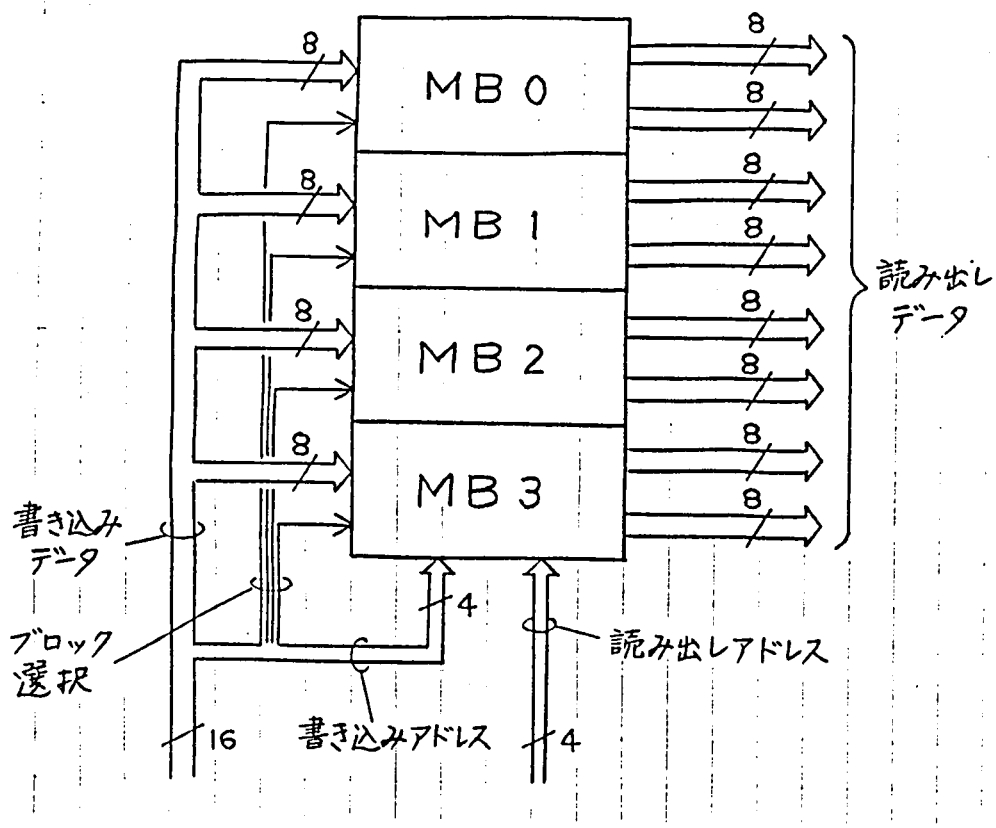


図３．８　メモリユニットの構成

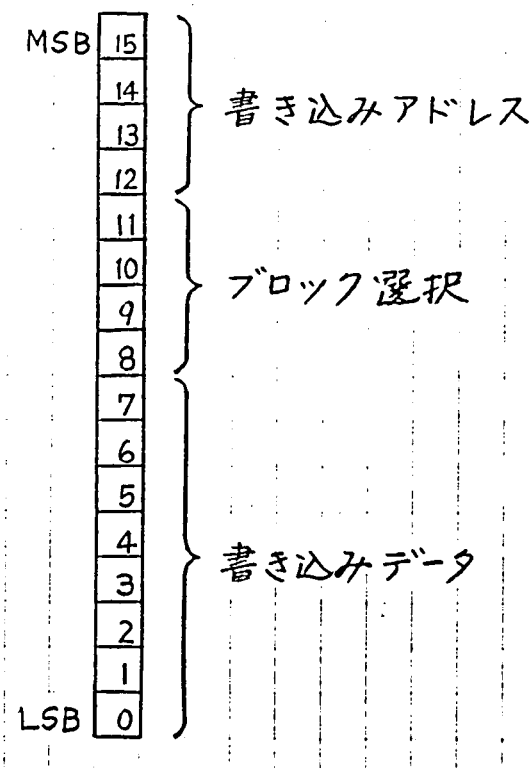


図３．９　メモリユニットへ書き込む際のリンケージ
ユニット入力バスのビット構成

３．３．２　演算回路の構成とその機能

第３．２．３項の検討を踏まえて決定した、ＩＳＰのプロセッサユニットとリンケージユニットの演算回路構成を、図３．１０に示す。また、それぞれの演算回路の機能を表３．４に示す。

図３．７に示すように、それぞれのＰＥには、データユニットやメモリユニットなどから、合計５バイトの演算データが与えられる。それらは、結局２バイトに選択（および結合）されて、ＰＥの前段のＡＬＵ(A)に入力される。ＡＬＵ(A)は、加減算のほか、ビット単位での論理演算と、入力データをそのまま出力するＮＯＰ機能を有している。

ＡＬＵ(A)の出力は、ＰＥの後段のＡＬＵ(B)に入力される。ＡＬＵ(B)は、乗算のほか、絶対値算出機能や、“High”レベルのビット数をカウントする機能、ＮＯＰ機能などを有している。ＡＬＵ(B)の演算結果は、１６ビットでリンケージユニットに出力される。

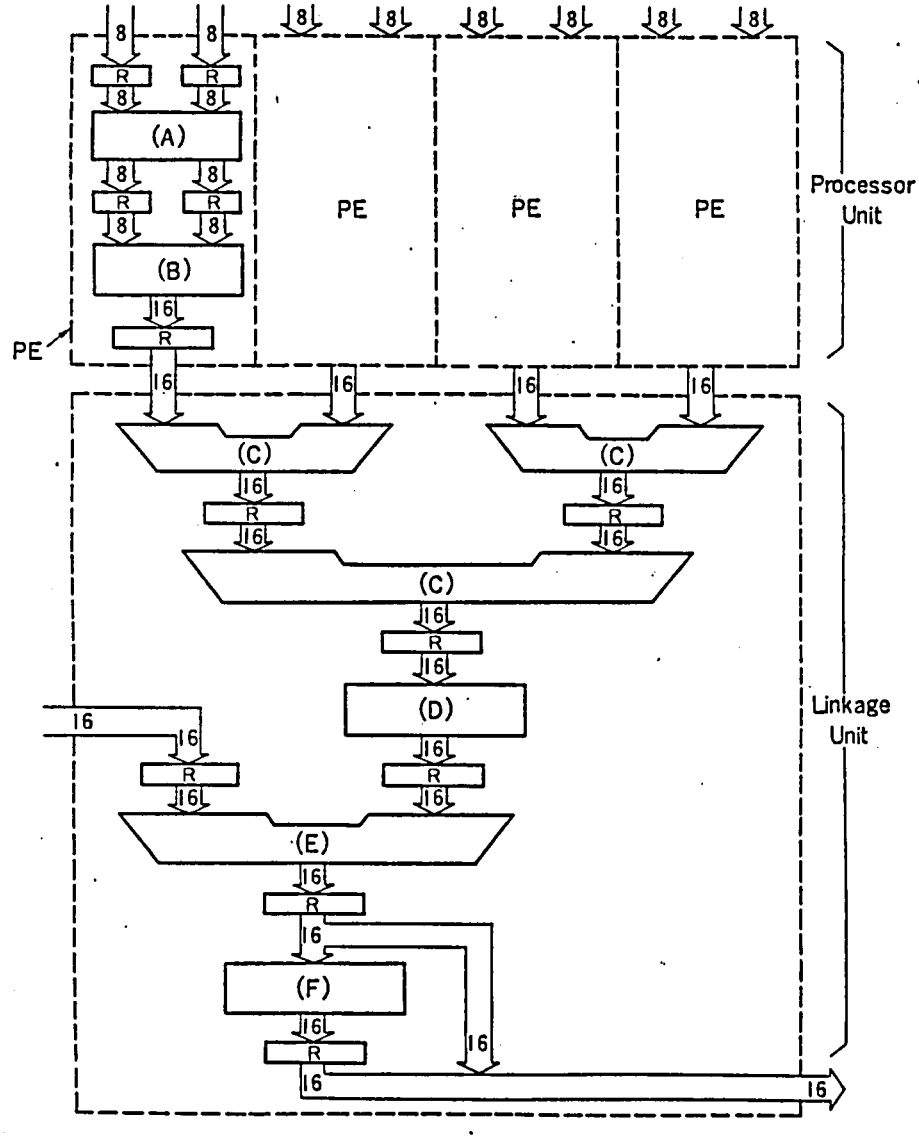


図３．１０　プロセッサユニットとリンケージユニットの演算回路構成

カーネルの形成に必要としないＰＥに対しては、ＡＬＵ(Ｂ)の出力はキャンセルされ、リンケージユニットでの演算に影響を与えないデータ（たとえば、加算に対する０）が出力される。

図２．２０に示すリンケージユニット内の４入力ＡＬＵは、図３．９では、三つのＡＵ(Ｃ)（Arithmetic Unit Ｃ）と一つのＡＵ(Ｄ)に分割されている。ＡＵ(Ｃ)は加減算などの機能を有し、ＡＵ(Ｄ)は絶対値算出のほか、2^n（ｎは０から８の整数）による除算の機能を有する。

図２．２０に示すリンケージユニット内の２入力ＡＬＵは、図３．１０では、ＡＵ(Ｅ)とＡＵ(Ｆ)の二つの演算回路に分割されている。ＡＵ(Ｅ)の機能はＡＵ(Ｃ)のそれと、ＡＵ(Ｆ)の機能はＡＵ(Ｄ)のそれと同じである。なお、特定の演算のためにチップ面積が増加するのを避けるため、ＳＱＲＴ機能は削除した。

以上述べたそれぞれの演算回路は、図３．１０に示すようにレジスタで分離されており、それぞれが一つのパイプラインステージを構成している。すなわち、プロセッサユニットとリンケージユニットで構成されるパイプライン段数は、合計８段になる。

表３．４　それぞれの演算回路の機能

演算回路	処　理　機　能			
A	ADD	SUB	AND	
	OR	EOR	ENOR	NOP
B	MUL	MAX	MIN	
	ABS*	NEG**	CNT***	NOP
C	ADD	SUB	MAX	MIN
D	ABS*	DIV	NOP	
E	ADD	SUB	MAX	MIN
F	ABS*	DIV	NOP	

* ABS: Absolute value
** NEG: Negative value
*** CNT: High-level-bit counting

３．４　ＩＳＰの処理機能

前節で論じたように、ＩＳＰは、画像データの入力方式の選択と、内部演算回路の機能の組み合わせとにより、多くの画像処理機能を実現することが可能となっている。しかも、これら多くの画像処理機能は、ＩＳＰの並列処理・パイプライン処理を妨げないため、256×256画素サイズ程度の画像ならば、ＴＶカメラからの信号転送速度（ビデオレート）で、高速に処理することができる。なお、種々の画像処理機能は、コントロールユニット内のプログラマブル制御レジスタの内容を変更するだけで選択できるため、汎用性のある画像処理システムの構築が可能である。

表３．５に、ＩＳＰの主な画像処理機能を示す。表３．５に示すように、ＩＳＰは、２値・濃淡・色彩画像に対して、種々の基本演算を実行することができる。以下、表３．５のそれぞれの処理機能について、簡単に述べる。

３．４．１　２値画像処理機能

２枚の画像を入力データとして、それらの画像の同じ（もしくは、重なる）位置にある画素間の論理演算は、ＩＳＰ内の１個のＰＥを用いて、１画素単位から最大８画素単位で実行できる。たとえば、図３．３の構成において、画像データをＡバスとＢバスから入力し、左端のＰＥのＡＬＵを用いて実行すればよい。

３×３画素のカーネルにおいて、１画素でも“１”（黒）があれば、３×３のカーネルの中央画素データを“１”に変える膨張（dilation）や、その逆に、１画素でも“０”（白）があれば、中央画素データを“０”に変える収縮（erosion）の処理は、ＩＳＰ内の３個のＰＥを用いることにより実行される。たとえば、図３．１の構成において、あらかじめ、それぞれのＲＡＭに$(11100000)_2$のデータを書き込んでおく。そして、そのデータとＡバスから入力される画像データとを、ＰＥで論理積（ＡＮＤ）を行った後、結果のビット列の中の“１”の総数を計算する。この“１”の総数が、０か否か、もしくは９か否かを、エバリュエーションユニットで確かめることにより実行できる。膨張処理では、“１”の総数が０の時、中央画素を“０”と変換し、１以上の時“１”とする。一方、収縮処理の場合、“１”の総数が９の時、中央画素を“１”とし、８以下の時“０”と変換する。

テンプレートを用いたパターンマッチングの処理は、ＩＳＰ１個で、８×４画素の処理が一度に実行できる。またカーネルは、ＩＳＰの追加や時分割処理により拡張できる。パターンマッチングについては、第５章において、ＩＳＰのシステム構築例で取り上げる。

３．４．２　濃淡画像処理機能

２枚の画像を用いる画像間の算術演算は、２値画像の画像間論理演算と同様に、ＩＳＰ内の１個のＰＥを用いて実行できる。たとえば、図３．３の構成において、ＡバスとＢバスから画像データの入力し、左端のＰＥのＡＬＵを用いて実行すればよい。

空間積和演算に対しては、１個のＩＳＰで、１×４画素の処理が可能である。ＩＳＰの追加や時分割処理によって、カーネルを拡張できる。この演算については、第５章において、ＩＳＰのシステム構成例で取り上げる。

１次微分オペレータとしての各種の非線形近傍演算は、複数回の空間積和演算に分解するか、もしくは、図３．３や図３．４の構成において実行できる。この演算についても、第５章において、ＩＳＰのシステム構築例で取り上げる。

画像の表示において、通常、個々の画素にはある面積を与えるが、演算の上では、画像平面上の一つの座標点で代表する。補間演算とは、濃度値が既知の座標（画素）以外の座標点の濃度値を、その座標点の近傍の、既知濃度値から算出する演算である。一座標点の補間を行うとき用いる濃度値が既知の画素の数によって、４点線形補間、９点２次補間、

表３．５　ＩＳＰの代表的処理機能

画　像	演　算
２値画像	・画像間論理演算（AND, OR, EOR, ENOR） ・膨張／収縮 ・パターンマッチング
濃淡画像	・画像間算術演算（Addition, Subtraction） ・空間積和演算（平滑，ラプラシアン，etc） ・非線形近傍演算 a b / c d ・$\|a-d\|+\|b-c\|$ ・$(a-d)^2+(b-c)^2$ ・$[\|a-b+c-d\|+\|a+b-c-d\|]/2$ A B C / D E F / G H I ・$\|A+B+C-G-H-I\|+\|A+D+G-C-F-I\|$ ・$\|E-A\|+\|E-C\|+\|E-G\|+\|E-I\|$ ・$\|E-B\|+\|E-D\|+\|E-F\|+\|E-H\|$ ・$\|A+2B+C-G-2H-I\|+\|A+2D+G-C-2F-I\|$ ・$\|\max(A,B,C,D,F,G,H,I)-E\|$ etc. ・４点線形補間演算 ・９点２次補間演算 ・16点 cubic 補間演算 ・残差検定法 ・固定２値化処理 ・浮動２値化処理 ・擬似メディアン・フィルタリング ・$\max[\min(A,B,C),\ \min(D,E,F),\ \min(G,H,I)]$ ・$\min[\max(A,B,C),\ \max(D,E,F),\ \max(G,H,I)]$
色彩画像	・原色間演算（ex. $\|R-G\|+\|G-B\|+\|B-R\|$） ・濃度変換（$\alpha R+\beta G+\gamma B$） ・色彩系変換 $\begin{bmatrix}R\\G\\B\end{bmatrix}=\begin{bmatrix}\ \ M\ \ \end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix}$ ・色彩距離演算 ・$\|R-\alpha\|+\|G-\beta\|+\|B-\gamma\|$　・$(R-\alpha)^2+(G-\beta)^2+(B-\gamma)^2$

１６点 cubic convolution 補間などが提案されている[16,17]。図３．１１は、入力画像ｆ(ｘ)に何らかの変換処理を施して、出力画像ｇ(ｙ)を生成する例を示し、また、図３．１２は、出力画像の一画素であるy_{22}の濃度値の求め方を模式している。

図３．１２において、縦方向の線分は、その座標点の濃度値を表している。図３．１２(ａ)の最近傍近似においては、画素y_{22}に最も隣接している画素x_{22}の濃度値で近似している。図３．１２(ｂ)の４点線形補間演算では、画素y_{22}の周辺の４個の画素の濃度値を、直線で補間して求めている。図３．１２(ｃ)の９点２次補間演算においては、画素y_{22}の最近傍画素x_{22}を中心とする９個の画素の濃度値を、２次曲線を用いて補間している。図３．１２(ｄ)の１６点 cubic convolution 補間演算では、画素y_{22}に隣接する１６個の画素の濃度値を、sinc 関数で補間して求めている。

これらの補間演算は、濃度値が既知の画素までの距離に依存する係数と、その画素の濃度値との積を総和する積和演算で実行できる。たとえば、４点線形補間演算における濃度値の算出式は、表３．２のように表すことができる。ＩＳＰにおいては、それぞれの画素までの距離に依存する係数を、荷重係数としてメモリユニットのＲＡＭに記憶しておき、近傍４画素の濃度値を入力データとする積和演算を実行することによって、この補間演算を実行できる。この時必要なＰＥ数は近傍画素数と同じ４個なので、図３．５の構成で動作させるならば、必要なＩＳＰは１個でよい。しかしメモリユニットは、個々のＰＥに対してそれぞれ１６バイトの記憶容量しか持たないので、１６通りの荷重係数しか記憶できず、その結果、ＩＳＰ１個では、近傍４画素内を１６個の座標点で近似して補間せざるをえない。そこで、よりきめ細かな補間演算を実行するには、複数個のＩＳＰを用いることになる。たとえば、４個のＩＳＰを用いれば、６４通りの荷重係数を記憶でき、近傍４画素内を６４個の座標点で近似して補間することが可能になる。

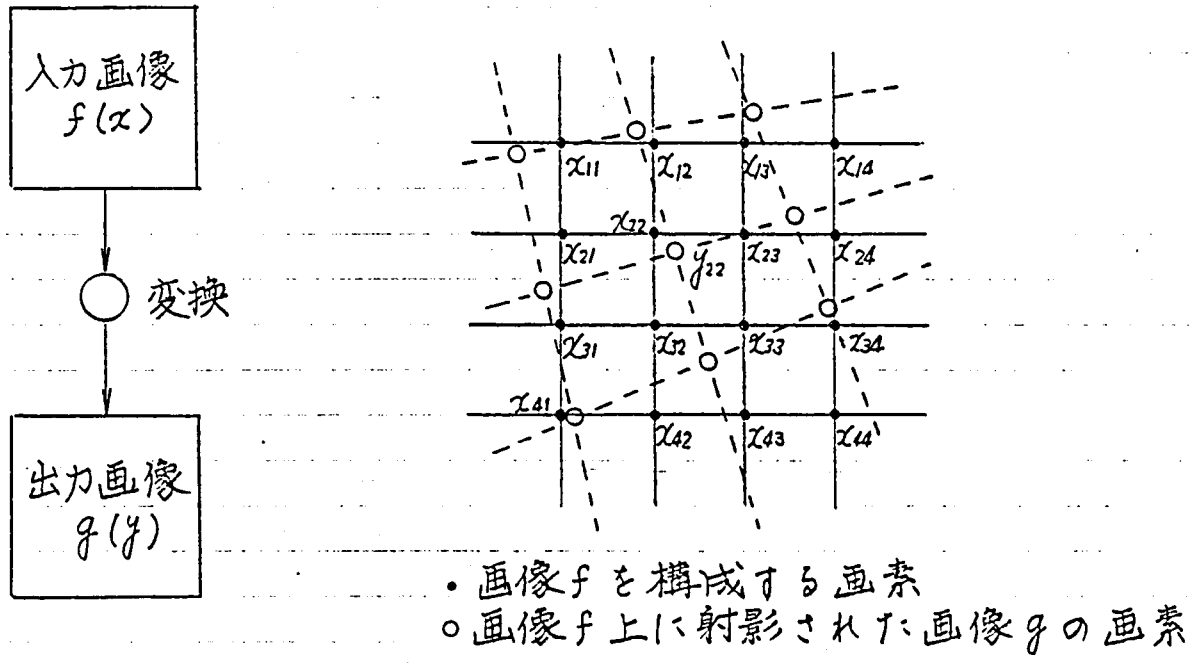
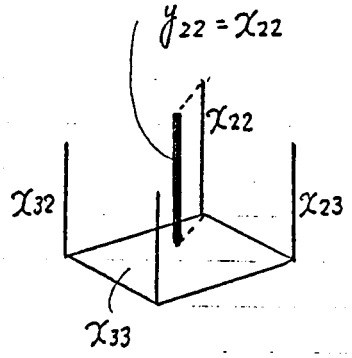
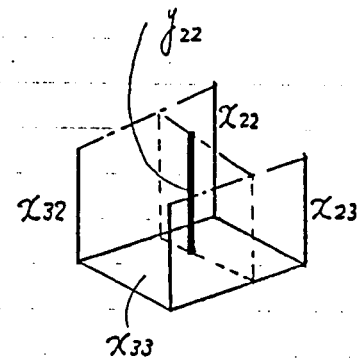


図３．１１　画像変換の例

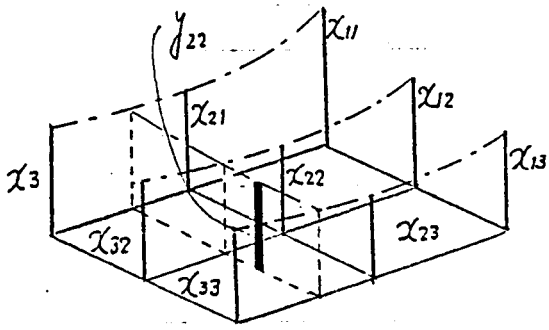


（a） 最近傍近似



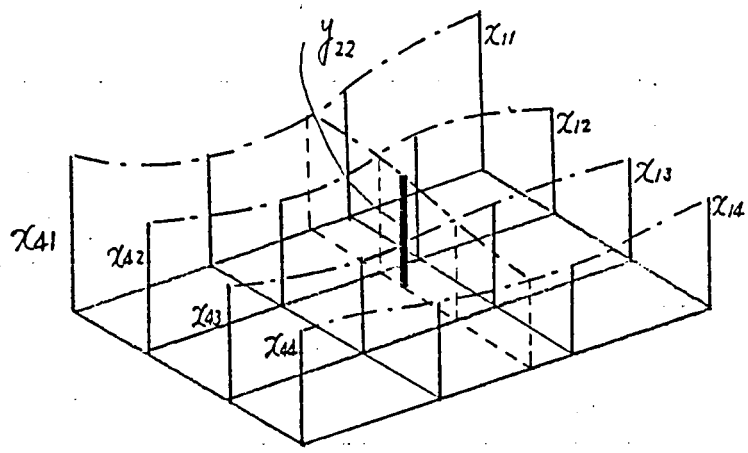
——·——·—— は直線

（b） 4点線形補間



——·——·—— は2次曲線

（c） 9点2次補間



——·——·—— は sinc 関数による補間曲線

（d） 16点 cubic convolution 補間

（a）～（d）で、縦の線分はそれぞれの画素の濃度値を示す。

図3．12 補間演算

残差検定法とは、濃淡画像におけるテンプレート・パターンマッチングであり、その演算式は、表３．２のように示される。演算オペレーションが積和と残差（画素間の濃度差の総和）で異なるほか、システム構成およびカーネル拡張などについては、空間積和演算と同じである。

２値化処理は、次章で詳しく論ずるエバリュエーションユニット内の比較回路により実行される。固定２値化処理は、エバリュエーションユニットの閾値レジスタに閾値を設定することで実行できる。一方、浮動２値化処理は、いくつかの近傍画素の濃度値の平均と、対象画素の濃度値との差を固定２値化処理することで実行できる。つまり、図３．４の構成において、近傍画素データをＡバスに、対象画素データをＢバスに入力し、それぞれのＰＥで減算を実行させ、リンケージユニットで（加算＋シフト除算により）平均値を求める。その結果をエバリュエーションユニットで、固定２値化することになる。近傍画素数が５個以上の時は、複数個のＩＳＰを用いることになる。

また、擬似メディアンフィルタリング処理とは、まず、３×３のカーネルのそれぞれの行について最大（もしくは最小）の濃度値を持つ画素データを抽出し、そのデータ間において最小（もしくは最大）の濃度値で、カーネルの中央の画素データを変換する演算である。この場合、ＰＥでは演算せず、すべての処理はリンケージユニットにおいて実行される。カーネルの拡張は、ＰＥ増設方式においてもＰＥ節約方式において可能である。

３．４．３　色彩画像処理機能

色彩画像は、複数の濃淡画像から成っているため、個々の濃淡画像でみれば、前項で述べた濃淡画像処理機能のすべてを適用できる。ここでは、色彩画像に固有の処理機能について述べる。

色彩画像を、光の三原色である赤・緑・青に分解すると、それぞれの画素について、三色の濃度値を用いての演算ができる。濃淡演算の処理機能である積和演算や残差検定法などを応用することにより、濃度変換処理や色彩空間上での距離演算などが実行できる。たとえば表３．２に示す色彩距離演算は、図３．５の構成において、ＰＥを３個用いて実行できる。ＩＳＰのＲＡＭには、あらかじめ標準色の３原色値、α、β、γを書き込んでおく。色彩画素データは、Ａ、Ｂ、およびＬＩバスから入力され、ＰＥにおいて、ＲＡＭから読み出されたデータとの差および自乗が計算される。その結果がリンケージユニットで加算され、出力データとなる。（この数値の平方根が最終結果であるが、平方根の機能はＩＳＰにはない。）

また、複数の積和演算で実行できるマトリックス演算により、色彩系の変換処理も可能である。たとえば、ＣＩＥ（国際照明委員会）で規定しているＸＹＺ表色系は、ＲＧＢ表色系から下記の演算式で求められる[17]。

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.62 & 0.17 & 0.18 \\ 0.31 & 0.59 & 0.11 \\ 0 & 0.066 & 1.02 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

この演算は、３個のＩＳＰを用いて実行できる。それぞれのＩＳＰは、図３．５の構成において、互いに独立してＸ、Ｙ、およびＺの値を計算することになる。荷重係数は、あらかじめＲＡＭに格納しておき、Ｒ、Ｇ、およびＢの濃度値は、それぞれＡ、Ｂ、およびＬＩバスから入力され、ＰＥにおいて乗算が、リンケージユニットにおいて加算が実行される。

３．５　まとめ

一般産業のさまざまな分野に、画像処理用ＬＳＩ－ＩＳＰが適用できるよう、２値・濃淡・色彩画像に対する種々の基本的な局所近傍演算を実行できる、多機能化を検討した。前章で論じたように、ＩＳＰの基本アーキテクチャは、濃淡画像の最も基本的な処理である空間積和演算を題材として決定されている。本章では、この基本アーキテクチャの上で、種々の基本演算を実行することを論じた。

基本アーキテクチャからは、次の点が明らかである。

（１）　プロセッサユニットのＰＥは、画素データを変換する機能を担う。

（２）　リンケージユニットの４入力ＡＬＵは、変換されたデータを統合する機能を担う。

（３）　リンケージユニットの２入力ＡＬＵは、統合されたデータを再統合する機能を担う。

そこで、これら三つの構成的機能を念頭に置いて、まずＬＳＰ内の構成要素の有機的な結合に着目した構造的アプローチを採り、次にそれぞれの構成要素に持たせる機能内容に着眼した機能的アプローチを採った。

構造的アプローチにおいては、プロセッサユニットのＰＥへのデータ供給方式に検討を加え、次の五つのデータ供給方式を採用した。

（１）　それぞれのＰＥに、８ビットの画像データと８ビットの演算オペランドを与える。画像データはＳＲを介して、演算オペランドはＲＡＭから供給される。（図３．１）

（２）　それぞれのＰＥに、１６ビットの画像データが与えられる。画像データは８ビットずつに分けられ、いずれもＳＲを介して供給される。（図３．２）

（３）　それぞれのＰＥに、１６ビットの画像データが与えられる。画像データは８ビットずつに分けられ、一方はＳＲを介して、他方は直接すべてのＰＥ共通に供給される。（図３．３）

（４）　それぞれのＰＥに、８ビットの画像データと８ビットの演算オペランドを与える。画像データはそれぞれのＰＥ別々に、ＬＳＩ外部から直接供給される。演算オペランドはＲＡＭから供給される。（図３．４）

（５）　それぞれのＰＥに、８ビットの画像データと８ビットの演算オペランド、さらに８ビットのカーネル指定データが与えられる。画像データはＳＲを介して、演算オペランドとカーネル指定データはＲＡＭから供給される。（図３．５）

機能的アプローチにおいては、まず２値・濃淡・色彩画像に対する種々の基本演算を、個々の演算オペレーションのレベルで、画素データ変換機能、変換データ統合機能、統合データ再統合機能の三つの構成的機能に分割した。そして、分割された演算オペレーションを実行できるよう、プロセッサユニットとリンケージユニットの演算回路の構成と、それぞれの機能を次のように決定した。（図３．９）

（１） プロセッサユニットのＰＥは、加減算を主な機能とするＡＬＵ(A)と、乗算を主な機能とするＡＬＵ(B)から構成する。

（２） リンケージユニットの４入力ＡＬＵは、加減算を主な機能とする三つのＡＵ(C)と、絶対値算出機能を有する一つのＡＵ(D)から構成する。

（３） リンケージユニットの２入力ＡＬＵは、加減算を主な機能とするＡＵ(E)と、絶対値算出機能を有するＡＵ(F)から構成する。

これらの結果、並列処理・パイプライン処理を併合した基本アーキテクチャ上で、２値・濃淡・色彩画像に対する種々の基本演算の実行が可能となった。

３．６　第３章の参考文献

[1]福島　忠、小林芳樹、平沢宏太郎、坂東忠秋、江尻正員、柏岡誠治、桑原　洋：画像処理用ＬＳＩ－ＩＳＰの開発、情報処理学会第２６回全国大会講演論文集、pp.939～940 (1983).

[2]福島　忠、小林芳樹、平沢宏太郎、坂東忠秋、江尻正員：画像処理用ＬＳＩ－Image Signal Processor のアーキテクチャ、電子通信学会論文誌(C)、Vol. J66-C, No. 12, pp.959～966 (1983).

[3]福島　忠、小林芳樹、平沢宏太郎、坂東忠秋、柏岡誠治、加藤　猛：多機能性を実現する画像処理用ＬＳＩ－ＩＳＰのアーキテクチャ、情報処理学会論文誌、Vol. 25, No. 5, pp.728～735 (1984).

[4]Fukushima, T., Kobayashi, Y., Hirasawa, K., Bandoh, T., Ejiri, M. and Kuwahara, H. : An Image Signal Prosessor, IEEE ISSCC Digest of Technical Papers, Vol.26, pp.258～259 (1983).

[5]Roberts, L. G. : Machine Perception of Three Dimensional Solids, in Tippett, J. T., et al. (ed.) Optical and Electro-Optical Information Proceesing, MIT Press, p.159 (1965).

[6]Forsen, G. E. : Processing Visual Data with an Automaton Eye, Cheng, G. C., et al. (ed.) Pictorial Pattern Recognition, Thompson, Washington, p.471 (1968).

[7]Rosenfeld, A. and Thurston, M. : Visual Texture Analysis 2, University of Maryland Computer Science Center Technical Report 70-129 (1970).

[8]Ejiri, M., Uno, T., Yoda, H., Goto, T., and Takeyasu, T. : A Prototype Intelligent Robot That Assembles Objects from Drawings, IEEE Transactions, Vol. C-21, No. 2, p.161 (1972).

[9]Kelly, M. D. : Edge Detection in Pictures by Computer Using Planning, in Machine Intelligence, Edingburgh Uiniversity Press, No. 6, p.397.

[10]Rosenfeld, A. : Pictuer Processing by Computer, Academic Press, New York (1969).

[11]長尾　真、金出武雄：パターン認識における縁・線の抽出、電子通信学会誌、Vol. 55, No. 12, pp.1618～1627 (1972).

[12]長尾　真：画像処理論、コロナ社 (1983).

[13]福島　忠、加藤　猛：画像処理用ＬＳＩ－ＩＳＰとその応用、O plus E、No. 5, pp.76～86 (1984).

[14]柏岡誠治、江尻正員、坂本雄三郎：時分割パターン認識技術による群制御トランジスタ組立システム、電気学会論文集(C)、Vol. 96, No. 1, pp.9～16 (1976).

[15]Fukushima, T. : Image Signal Processor Computes Fast Enough for Gray-Scale Video, Electronic Design, Vol.32, No.20, pp.209～215(1984).

[16]井原廣一、山本康成：地球観測衛星（ランドサット）画像の精密補正技術、電気学会雑誌、Vol. 101, No. 4, (1981).
[17]田村秀行（監修）：コンピュータ画像処理入門、総研出版（1985).

第４章　マルチマスクオペレーションへの対応

４．１　まえがき

画像処理の一般産業への実用化を目指した画像処理用ＬＳＩ－ＩＳＰは、①ビデオレートでの高速処理、②任意の大きさへのカーネルの拡張性、③各種の基本演算を実行できる多機能性、を目標とした[１，２]。①の高速処理は、並列処理とパイプライン処理により、②の拡張性は、ＰＥ増設方式とＰＥ節約方式の二つのカーネル拡張方式により、③の多機能性は、ＰＥへのデータ供給方式と演算回路の機能分割により、それぞれ実現した[３，４]。

しかし、実際の応用面からみると、画像処理システムの一素子であるＬＳＩには、上記の三つとは異なった角度からの要望がある。それは、いかに経済的なシステムの構築を可能にするか、という点である。第２章、第３章で検討された内容を持つＬＳＩで処理可能であっても、経済性を備えなければ実際的ではない。

第３章では、画像処理ＬＳＩの多機能性について論じたが、空間積和演算やパターンマッチングなどのように、演算オペランドとしてのマスクデータを用いるマスク演算も数多い。ここでいうマスクデータとは、空間積和演算における荷重係数や、パターンマッチングにおける標準図形としてのテンプレートなどである。

しかし、これまでの議論では、切り出したカーネルと演算するマスクデータは、１個であることが前提であった。ところが、２個以上のマスクデータを用いるアルゴリズムも、これまで数多く提案され、その有用性も実証されている。たとえば、８個の差分型マスクを用いて、その最大出力値からエッジの強度や方向を求めるものとして、Prewitt[５]、Kirsch[６]、Robinson[７]、などのテンプレート型オペレータがある[８]。また、半導体チップの自動ワイヤボンディング装置では、複数のテンプレートを用いた複合パターンマッチングがある[９]。複合パターンマッチングでは、±15° 程度の回転ずれを許容する、位置検出を実現している。

このような複数個のマスクデータを用いるマスク演算――以下、マルチマスクオペレーション(multi-mask operation)と呼ぶ――は、いくつかの単一マスクによる演算と、その結果間の演算に分割すれば、充分に処理可能である。しかし、それが必ずしもシステムの経済性に結びつくとは限らない。この場合、画像メモリ参照のオーバーヘッド、演算回路の付加などを考えると、画像処理用ＬＳＩにワンチップ化できるのが好ましいといえる。

以上の観点から、ここでは、これまで論じたＩＳＰのアーキテクチャを損うことなく、マルチマスクオペレーションを効率よく実行できる方法を考察し、アーキテクチャ上の改善を検討する。

４．２　マルチマスクオペレーションの処理方針

ある原画像からカーネルを形成する画素を切り出して、局所近傍演算を施す場合、通常マスクデータとのマスク演算となる。たとえば空間積和演算では、マスクデータとして荷重係数が用意され、切り出された画素データは、荷重係数と掛け合わされた後、加算される。この時、一つの画像処理で用いられるマスクデータが、平滑化オペレータやラプラシアンオペレータなどの単一のものと、Prewittオペレータのテンプレート型のような複数のものとに分けられる。ここでは、前者をシングルマスクオペレーション、後者をマルチマスクオペレーションと呼んで区別する。シングルマスクオペレーションのいくつかを図４．１に[12,13]、また代表的なマルチマスクオペレーションを表４．１に[５～９]、それぞれ示す。

表４．１の中で、Prewitt、Kirsch、Robinson、のテンプレート型オペレータはよく知られている。これらは、３×３のカーネルに８個の差分型マスクを掛け合わせ、その最大値をエッジの強度として、またその時のマスクがＭｉならば、ｉをエッジの方向として出力するものである。

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

0	0	0	1	1	1	0	0	0
0	0	0	1	1	1	0	0	0
0	0	0	1	1	1	0	0	0
1	1	1	-4	-4	-4	1	1	1
1	1	1	-4	-4	-4	1	1	1
1	1	1	-4	-4	-4	1	1	1
0	0	0	1	1	1	0	0	0
0	0	0	1	1	1	0	0	0
0	0	0	1	1	1	0	0	0

（ａ）　ラプラシアンオペレータ

1	1	1
1	1	1
1	1	1

（ｂ）　平滑化オペレータ

0	-1	0
-1	5	-1
0	-1	0

（ｃ）　高域強調オペレータ

図４．１　シングルマスクオペレーションの例

表４．１　代表的なマルチマスクオペレーション

入力画像	名　　称	内　　容	マ　ス　ク　デ　ー　タ
２　値	複合パターンマッチング	複数個の標準パターンと照合して，最もよく一致した標準パターンと，その時の一致度を求める	任　　意
濃　淡	Prewittオペレータ（テンプレート型）	M0～M7のマスクを3×3のカーネルに掛け合わせ，その最大値をエッジ強度とし，その時のマスクがMiならば，エッジ方向を i とする	M0: 1 1 1 / 1 -2 1 / -1 -1 -1 M1: 1 1 1 / 1 -2 -1 / 1 -1 -1 M2: 1 1 -1 / 1 -2 -1 / 1 1 -1 M3: 1 -1 -1 / 1 -2 -1 / 1 1 1 M4: -1 -1 -1 / 1 -2 1 / 1 1 1 M5: -1 -1 1 / -1 -2 1 / 1 1 1 M6: -1 1 1 / -1 -2 1 / -1 1 1 M7: 1 1 1 / -1 -2 1 / -1 -1 1
	Kirschオペレータ		5 5 5 / -3 0 -3 / -3 -3 -3 5 5 -3 / 5 0 -3 / -3 -3 -3 5 -3 -3 / 5 0 -3 / 5 -3 -3 -3 -3 -3 / 5 0 -3 / 5 5 -3 -3 -3 -3 / -3 0 -3 / 5 5 5 -3 -3 -3 / -3 0 5 / -3 5 5 -3 -3 5 / -3 0 5 / -3 -3 5 -3 5 5 / -3 0 5 / -3 -3 -3
	Robinsonオペレータ		1 2 1 / 0 0 0 / -1 -2 -1 2 1 0 / 1 0 -1 / 0 -1 -2 1 0 -1 / 2 0 -2 / 1 0 -1 0 -1 -2 / 1 0 -1 / 2 1 0 -1 -2 -1 / 0 0 0 / 1 2 1 -2 -1 0 / -1 0 1 / 0 1 2 -1 0 1 / -2 0 2 / -1 0 1 0 1 2 / -1 0 1 / -2 -1 0
	Prewittオペレータ（ディファレンシャル型）	エッジ強度およびエッジ方向を求める	SX: -1 0 1 / -1 0 1 / -1 0 1 SY: -1 -1 -1 / 0 0 0 / 1 1 1 (エッジ強度) $=\sqrt{SX^2+SY^2}$ or $\lvert SX\rvert+\lvert SY\rvert$ (エッジ方向) $=\tan^{-1}(SY/SX)$
	Sobelオペレータ		-1 0 1 / -2 0 2 / -1 0 1 -1 -2 -1 / 0 0 0 / 1 2 1
色　彩	色彩距離による分類	複数個の標準色と照合して，色彩距離がある許容範囲の時，その標準色として分類する	$D_i=\sqrt{(R-\alpha_i)^2+(B-\beta_i)^2+(G-\gamma_i)^2}$ or $\lvert R-\alpha_i\rvert+\lvert B-\beta_i\rvert+\lvert G-\gamma_i\rvert$ D_i ：クラス i の標準色からの色彩距離 R, B, G ：入力画素の光の３原色の各濃度値 $\alpha_i, \beta_i, \gamma_i$ ：クラス i の標準色の光の３原色の各濃度値

このようなマルチマスクオペレーションの処理方式は、次の二つの観点から分類できる。第一は、一つのマスクデータに関する演算の、全画面にわたっての実行を、マスクデータの数と等しい回数繰り返すのか、もしくは、一つのカーネルを切り出す毎に、用意したすべてのマスクデータとの演算の実行を、全画面にわたって繰り返すのか、という点である。第二は、個々のカーネルに対して、用意したすべてのマスクデータとの演算を終えて、マスクデータと同数の演算結果を求めた後、その演算結果間の演算を一度に実行するのか、もしくは、一つのカーネルを切り出し、一つのマスクデータとの演算を終了する毎に、マスク演算結果間の演算も実行してゆくのか、という点である。

上記の二つの観点の組み合わせから、マルチマスクオペレーションの処理形態は、次の４通りが考えられる。

（１） マスクデータの数と同数のフレームメモリを用意し、先ずマスクデータ毎に全画面の処理を実行し、演算結果をフレームメモリに記憶する。そして、その後フレームメモリ間の演算を実行する。

（２） １枚のフレームメモリを用意し、先ず一つのマスクデータの処理を全画面にわたって実行し、演算結果をフレームメモリに記憶する。そして、二つ目以降のマスクデータの演算は、カーネル毎に演算結果が求められるつど、その演算結果はフレームメモリの内容とさらに演算されて、その結果がフレームメモリの内容を更新する。

（３） マスクデータの数と同数の一時記憶レジスタを用意し、一つのカーネルについてすべてのマスクデータとの演算を実行し、結果を一時記憶レジスタに記憶する。そして、レジスタ間の演算を実行して、最終結果をフレームメモリに書き込む。その後、次のカーネルの演算へ移行する。

（４） 一時記憶レジスタを１個用意し、一つのカーネルについて最初のマスクデータとの演算を実行し、結果を一時記憶レジスタに記憶する。同じカーネルについての二つ目以降のマスクデータとの演算は、結果を求めるつど、一時記憶レジスタの内容と演算され、最終結果がフレームメモリに書き込まれる。その後、次のカーネルの演算へ移行する。

（１）から（４）の演算方式を、ハードウェア規模の上から比較すると、（１）の方法が、マスクデータと同数のフレームメモリが必要なため、最もハードウェアの負担が重いことが分かる。（２）の方法によると、ワーク用のフレームメモリを演算結果用のフレームメモリで兼ねるならば、余分な回路は全く不必要である。また、（３）および（４）の方法では、一時記憶用のレジスタ回路が余分に必要である。

一方、画像メモリ参照の点からみると、（１）および（２）の方法が、読み出し・書き込みともに最も頻繁である。これは、アドレス処理のためのメモリ周辺回路の負荷が重いことを意味する。しかし、（３）および（４）の方法によると、画像メモリの参照は、入力画像の読み出しを除くと、最終結果を書き込む時だけですむ。

そこで、ＩＳＰにおいては、カーネルを切り出す毎に、すべてのマスク演算を行なうと同時に、それらの演算結果間の処理まで行なって、一つのカーネルに対する最終結果まで求める（４）の方法を採用することにした。なぜなら、より効率的なシステムを構築するためには、ハードウェア規模が小さいことだけでなく、画像メモリの参照を最小限にするこ

とが望ましいからである。つまり上に述べた方法によると、入力画像の読み出しと最終結果の書き込みの、２回のメモリ参照でマルチマスクオペレーションを処理でき、メモリ参照によるオーバーヘッドを、最小限に抑えることができる。

なお、表４．１からも知れるように、マルチマスクオペレーションにおける、マスク演算結果間の処理の多くは、比較演算によりなされうる。そこで、エバリュエーションユニットに比較器を設け、各種の２値化処理と合わせて、マスク演算結果間の処理を行なわせることにした。比較演算で算出できないマスク演算結果間の処理については、チップ面積などの理由により、ＩＳＰで実現するのは断念した。

４．３　アーキテクチャへの考察

４．３．１　アーキテクチャ上の課題

前節で述べた方針にもとづいて、ＩＳＰの基本アーキテクチャに対する改善を検討してゆく上で、次の四つの課題があると考えられる。

（１）　画像データのＰＥへの供給方式。

（２）　マスクデータのＰＥへの供給方式。

（３）　マスク演算結果間の比較演算方式。

（４）　比較演算結果の出力方式。

（１）と（２）の課題は、ＰＥにおける演算の順序と、リンケージユニットにおける演算の制御に係わる問題である。一方、（３）と（４）の課題は、エバリュエーションユニットにおける演算に関する問題である。そのため、（３）と（４）の課題は、（１）と（２）の課題から切り離して考えることができる。なぜなら、前節で述べた方針にもとづくと、カーネル毎に算出されたそれぞれのマスクデータに対する演算結果は、リンケージユニットからエバリュエーションユニットへ順次供給され、（１）と（２）の方式が異なっても、転送のタイミングが多少違ってくるだけだからである。そこで、（１）と（２）の検討と、（３）と（４）の検討を分けて考察する。

ここで考慮しなければならないのは、二つのカーネル拡張方式への対処である。なぜなら、二つの拡張方式において、画像の走査方式が異なるからである。第２．４節で述べたように、ＩＳＰは、用いるＰＥの数を増やすＰＥ増設方式と、ＰＥを時分割に使用するＰＥ節約方式の、二つの方式によりカーネルを拡張できる。画像の走査は、ＰＥ増設方式ではラスタ走査が用いられ、ＰＥ節約方式ではスティック走査が用いられる[３]。

しかし、第２．４．２項で述べたように、ラスタ走査はスティック走査の特殊なもの－－スティック長が１のスティック走査－－であるから、スティック走査により走査された画像データを処理できれば、ラスタ走査により走査された画像データも処理できることになる。つまり、ＰＥ節約方式に対処できれば、ＰＥ増設方式にも対処できるといえる。そこで、ＰＥ節約方式によるカーネル拡張に対して、上記の（１）から（４）の課題を検討することにする。

４．３．２　データ供給に関する検討

ここでは、前節であげた課題の、（１）画像データのＰＥへの供給方式と、（２）マスクデータのＰＥへの供給方式について検討するが、討論を簡潔にするため、スティック長が２のスティック走査を用いて、２個のマスクデータを扱うマルチマスクオペレーションを実行する場合について考察する。このマルチマスクオペレーションを実行する際の、一つのＰＥとそれに関連する部分だけを抜粋した模式図を、図４．２に示す。

図４．２において、Ｐ１とＰ２は画素データを、Ａ１とＡ２は１個目のマスクデータを、Ｂ１とＢ２は２個目のマスクデータを表わしている。なお同図においては、空間積和演算を想定して演算機能を選択しているので、リンケージユニットのＡＬＵからは、演算結果として次の二つの値、Ｒ１とＲ２、が出力される。

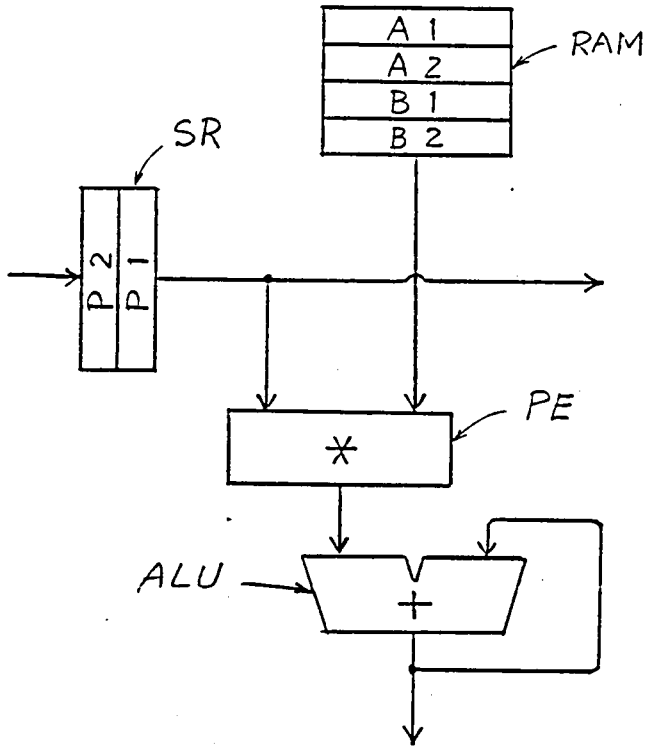


図４．２　スティック長２、マスクデータ数２の
マルチマスクオペレーションの模式図

$$R1 = A1 \times P1 + A2 \times P2.$$
$$R2 = B1 \times P1 + B2 \times P2.$$

ＰＥへの画像データとマスクデータの供給方式は、ＰＥにおける演算の実行順序に左右される。上記の演算では、一つのＰＥで４回の乗算が実行されなければならないが、次の二つの順序が考えられる。

（１）　Ａ１×Ｐ１，Ａ２×Ｐ２，Ｂ１×Ｐ１，Ｂ２×Ｐ２．

（２）　Ａ１×Ｐ１，Ｂ１×Ｐ１，Ａ２×Ｐ２，Ｂ２×Ｐ２．

つまり、(１)は、マスクデータ毎に演算を進める方式であり、(２)は、画像データ毎に演算してゆく方式である。

(１)の方式では、マスクデータ単位の演算を、マスク数だけ繰り返すことになる。そのため、演算の制御は容易であるが、同じ画素データを繰り返し入力する必要がある。この時ＳＲ（Shift Register）の遅延段数は、(スティック長)×(マスク数)となる。図４．２の例では、ＳＲ１個あたり４段にすればよい。しかし、データユニットには４個のＳＲがあることや、スティック長やマスク数の増加を考慮すると、ＳＲのゲート数は膨大なものになる。たとえば、表４．１にあるPrewittオペレータのテンプレート型を処理する場合、それぞれのＳＲの遅延段数は２４段になる。

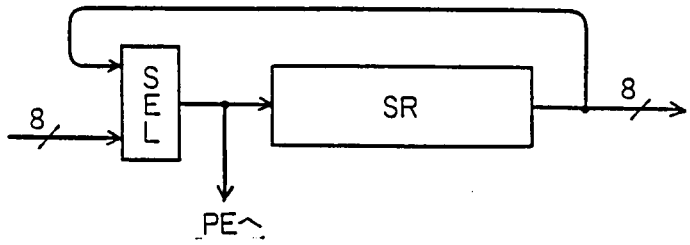


図４．３　シフトレジスタにおける
フィードバック機構

（２）の方式では、個々の画素データは、マスク数に等しい回数だけ続けて使用される。そのため、何度も入力する必要はない。マスク数が増加すれば、それに応じてＳＲの転送レートを低下すればよいので、ＳＲの遅延段数は、常にスティック長と同数でよい。しかし、それぞれのマスクデータに対する演算結果を算出するためには、リンケージユニットに、マスク数と同数の一時記憶バッファを用意して、複雑なデータ制御を施さなければならない。

（１）と（２）を比較すると、（１）の方がはるかに制御が容易である。一方、ハードウェア規模では、（１）は多段数のＳＲが必要であり、（２）は多数個の一時記憶バッファが必要である。この点ではいずれの方法が優れているともいえない。しかし、（２）の方法では、第２章で決定した基本アーキテクチャに、修正を加えなければならない。そこで、基本アーキテクチャを修正する必要のない（１）の方法において、ハードウェア規模を削減することを検討した。

（１）の処理手順を考察すると、画像データは常にサイクリックにＰＥに供給されていることが分かる。つまり、画像データをループ状にフローさせれば、サイクリックなデータ供給は可能である。そこで、ＳＲには、図４．３に示すようなフィードバック機構を設けることにした。つまり、ＰＥへ転送される画素データは、同時にＳＲの初段へフィードバックされ、マスク数に応じて再度ＰＥへ転送されることになる[11]。

なお、それぞれの画素データには、２ビットのシーケンスコード(sequence code)を付加して、その画素データがスティック内のどこに位置するか、という情報を持たせることにした。このシーケンスコードが、ＳＲのフィードバック機構や、リンケージユニットにおける統合演算を制御することになる。これらの結果、それぞれのマスクデータに対する演算結果は、スティック長に等しいマシンサイクル毎に、リンケージユニットから出力されることになる。

４．３．３　比較演算に関する検討

第４．２節の方針にもとづくと、エバリュエーションユニットは、比較器を中心とした構成となる。そこで、エバリュエーションユニットの構成は、比較器を用いる閾値処理(thresholding)を基本として、マルチマスクオペレーションにおけるマスク演算結果間の、比較演算を実現することから検討した。

閾値処理には、閾値が固定されたままの固定閾値処理と、処理の進行にともなって閾値の変化する浮動閾値処理とがある。ここでは、浮動閾値処理には、カーネル内の演算と固定閾値処理などに変換できるものもあるので、ＩＳＰにおいては、固定閾値処理についてのみ考察することにした。

固定閾値処理においては、参照データは固定の閾値により２値化されるが、より汎用性を高めるために、二つの閾値Ｔi、Ｔj(Ｔi＜Ｔj)、を用いた４種類の２値化処理を実現することにした。参照データをｆ、出力結果をｇとすると、４種類の２値化処理は次のように表わされる。

（ａ）　ｇ＝１　if　ｆ≦Ｔi　　, otherwise　ｇ＝０．
（ｂ）　ｇ＝１　if　Ｔj＜ｆ　　, otherwise　ｇ＝０．
（ｃ）　ｇ＝１　if　ｆ≦Ｔi　or　Ｔj＜ｆ　, otherwise　ｇ＝０．
（ｄ）　ｇ＝１　if　Ｔi＜ｆ≦Ｔj　　, otherwise　ｇ＝０．

これらを実行するためには、２個の比較器と２個の閾値レジスタが必要であるとともに、任意の閾値を用いられるよう、演算の実行に先立って、閾値レジスタの内容を自由に書き換えられる必要がある。さらに、比較結果により(たとえばｇ＝１の時)、参照データで閾値レジスタの内容を書き換えられるならば、参照データの最大値や最小値を求めることができる。また、２値化情報と合わせることにより、最大値もしくは最小値の座標値を求めることも可能である。

一方、マルチマスクオペレーションにおける、マスク演算結果間の比較演算を実現するには、次の二つの機能が要求される。

（１）　マスク演算結果のうちの最大値、もしくは最小値を、カーネル毎に摘出する。
（２）　（１）で求めた最大値、もしくは最小値を生み出したマスクデータの番号を作り出す。

これらのことを踏まえて、エバリュエーションユニットの構成を、図４．４に示すように決定した。

図４．４の構成において、リンケージユニットから与えられる参照データは、２個の比較器ＣＯＭＰ(comparator)に同時に入力される。それぞれの比較器は、通常、閾値レジスタＴＲ(thresholding register)の内容と参照データとを比較し、大小の判定結果を出力する。判定結果は、最終的には１ビットの２値データとして、ＬＳＩ外部に出力される。また比較器からは、比較した二つのデータの大きい方(もしくは小さい方)の値を、閾値レジスタに戻す。閾値レジスタは書き換え可能になっており、機能に応じて比較器の出力データで更新される。二つの比較器の一方は大きい値を、他方は小さい値を抽出するよう機

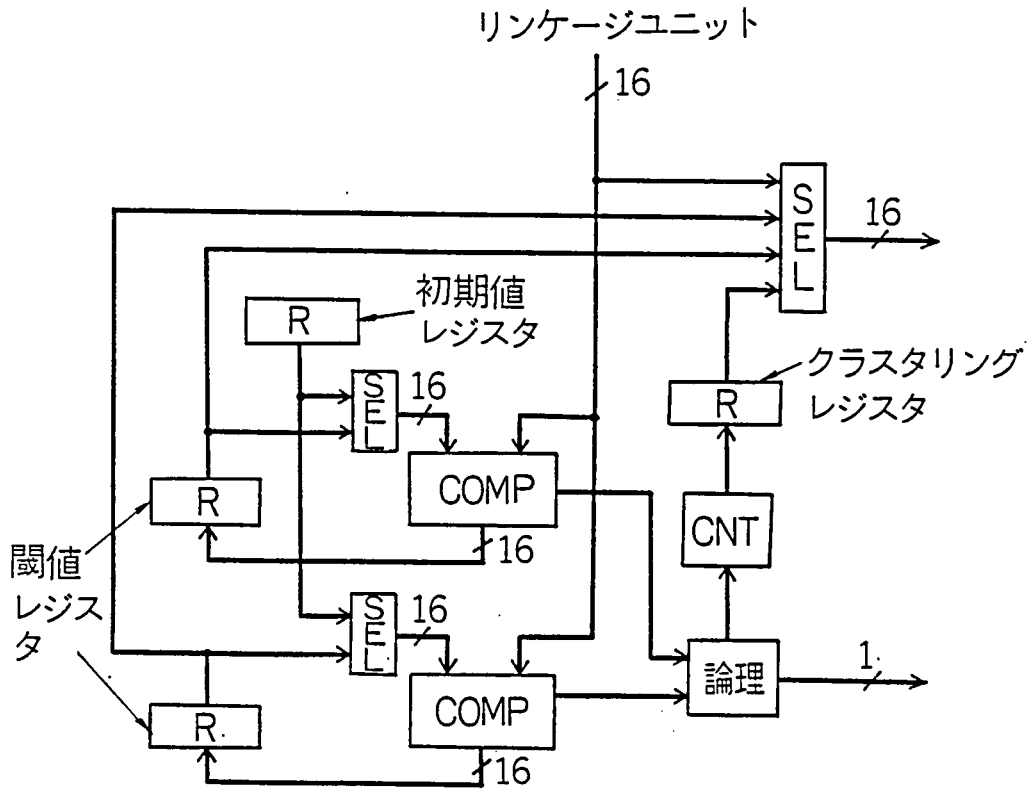


図４．４　エバリュエーションユニットの構成

能が固定されているので、閾値レジスタの書き替えを停止することにより、前述した４通りの固定閾値処理を実行できる。また、閾値レジスタの書き換えを許すことにより、参照データの最大値や最小値を求めることもできる。

しかし、これだけでは、(１)のカーネル毎にマスク演算結果間の、最大値・最小値を求めることはできない。なぜなら、カーネルが移動する毎に、閾値レジスタを初期化しなければならないからである。一方、一つのカーネルにおけるマスク演算結果間の処理が終了すると、その結果をＬＳＩ外部へ出力しなければならない。そのため、処理結果の出力が終了するまで、閾値レジスタを初期化できない。つまり、閾値レジスタの内容の出力と初期化のために、一つのカーネルについての処理を終了した後、次のカーネルについての処理の開始を遅らせる必要が生ずる。この問題を解決するために、図４．４に示す初期値レジスタＩＲ (initial register) を設けた。

新たなカーネルの最初の参照データが比較器に入力される時、閾値レジスタに代って、初期値レジスタの内容が比較器に与えられる。この時、いずれか一方の閾値レジスタの内容が、ＬＳＩ外部に出力される。閾値レジスタの内容が出力された後、初期値レジスタと参照データとの比較結果が、閾値レジスタに書き込まれる。

このように、初期値レジスタを設けて、閾値レジスタの初期化を省略することにより、新たなカーネルについての処理の開始を、遅延させなくてよくなった。なお、図４．４では繁雑を避けるため記入していないが、二つの閾値レジスタと初期値レジスタは、リンケージユニットの入力バスを介して、演算実行に先立って、任意の値に設定できる。

マルチマスクオペレーションのマスク演算結果間の処理を実現するには、前述の(２)の機能、すなわち、求めた最大値・最小値を生み出したマスク番号を、生成できなければならない。これを生成するのが、図４．４に示すバイナリィカウンタＣＮＴ（binary counter）である。バイナリィカウンタは、カーネルが移動する度に初期化され、マスク演算結果が比較器に与えられる毎にカウントアップする。バイナリィカウンタの内容は、比較器の判定結果により、適宜クラスタリングレジスタＣＲ（custering register）に書き込まれる。

これらの結果、一つのカーネルにおけるマスク演算処理が終了した時、マスク演算結果の最大値もしくは最小値は、閾値レジスタに、その値に関するマスク番号は、クラスタリングレジスタに残される。これらの値は、リンケージユニットの出力バスを介して、時分割でＬＳＩ外部に出力される。なお、図４．４において、閾値レジスタと初期値レジスタの出力選択や、バイナリィカウンタの初期化のタイミングなどの制御には、データユニットのＳＲや、リンケージユニットの統合演算の制御と同様に、それぞれの画素データに２ビットずつ付加されて入力される、シーケンスコードを用いている。

４．４　ＩＳＰのタイミング制御

第２章および第３章で述べたように、画像処理用ＬＳＩ－ＩＳＰは、４個のＰＥによる並列処理方式に加えて、多段演算回路によるパイプライン処理方式を採用している。この基本アーキテクチャ上で、マルチマスクオペレーションを実行するには、本章で検討したデータユニットとエバリュエーションユニットのタイミング制御を、いかに行うかという問題を解決しなければならない。

一般に、タイミングの制御方式には、ＬＳＩの外部からのタイミング信号に同期させる外部同期方式と、ＬＳＩ内部で生成したタイミング信号に同期させる内部同期方式とがある。前者の外部同期方式の場合、ＬＳＩ内部の制御を簡素化して、テスタビリティを向上できるという長所がある反面、周辺回路の負荷が大きくなるという短所がある。一方、後者の内部同期方式の場合、周辺回路を軽減できる反面、ＬＳＩの制御回路が複雑になる。特に、２個以上のＬＳＩを用いる場合の制御が繁雑である。そこで、ＩＳＰでは、ＬＳＩを追加してカーネルを拡張するのが容易であることを優先させ、外部同期方式を採用することにした。

マルチマスクオペレーションを実行する際に、タイミングの制御が必要とされるのは、次の２点である。一つは、データユニットのシフトレジスタにおける、画像データのフィードバック動作である。もう一つは、エバリュエーションユニットの２個の比較器における、比較演算動作である。これらの動作のタイミング制御を考察すると、次のように要約できる。

先ず、データユニットのシフトレジスタにおいては、画像データのフィードバックの開始と終了の情報が必要である。仮りに、図４．３において、制御信号が“１”の時、シフトレジスタの出力データがフィードバックされるとすると、図４．５に示すような制御信号が与えられなければならない。（ただし、図４．５は、スティック長が４、マスク数が３のマルチマスクオペレーションの場合を示す。）しかし、この制御信号は、立ち上がりと立ち下がりのタイミング情報が与えられれば、生成することができる。

一方、図４．４に示すエバリュエーションユニットにおいては、①比較器の演算実行タイミング、②比較器の入力セレクタの切り換えタイミング、さらに、③出力セレクタの切り換えタイミングを制御しなければならない。①の比較器の演算実行は、スティック１個につき１回であり、②の比較器の入力セレクタについては、最初のマスクについての演算時のみ、初期値レジスタを選ぶことになる。また、③の出力レジスタについては、第４．３．３項で述べたように、比較器の入力セレクタが初期値レジスタを選択している時に、比較器の出力レジスタの一方を選択しなければならない。つまり、図４．６に示すような制御を施さねばならない。（ただし、図４．６は、スティック長が４、マスク数が３のマルチマスクオペレーションの場合を示す。同図において、演算制御信号が“１”の時、比較器は比較演算を実行し、セレクタ制御信号が“１”の時、比較器の入力セレクタは初期値レジスタを、出力セレクタは比較レジスタのいずれか一方を選択するものとする。）

また、マルチマスクオペレーションの実行に限らず、種々のスティック長の処理を行うために、リンケージユニット内の２入力ＡＬＵにおける、アキューミュレーション動作のタイミングの制御も必要である。リンケージユニットの２入力ＡＬＵにおいては、ＬＳＩ

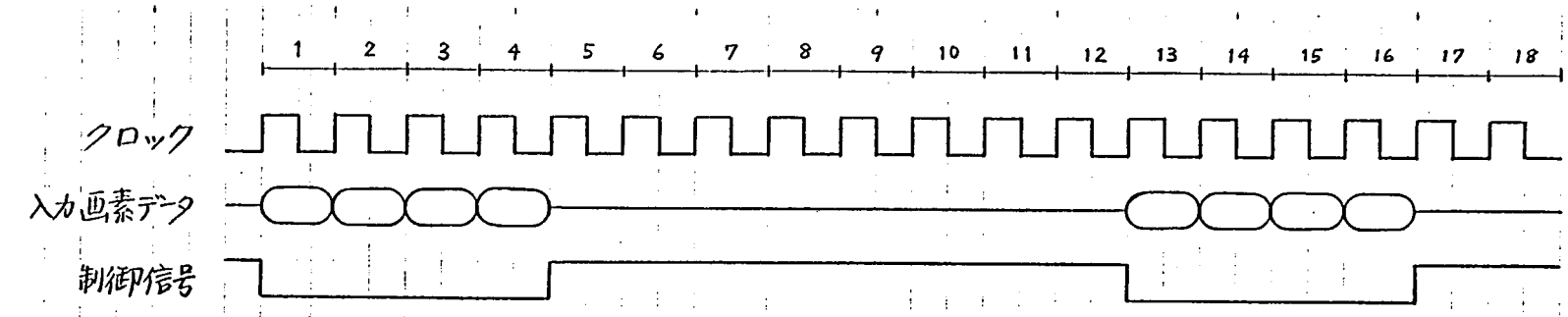


図４．５　データユニットのタイミングチャート

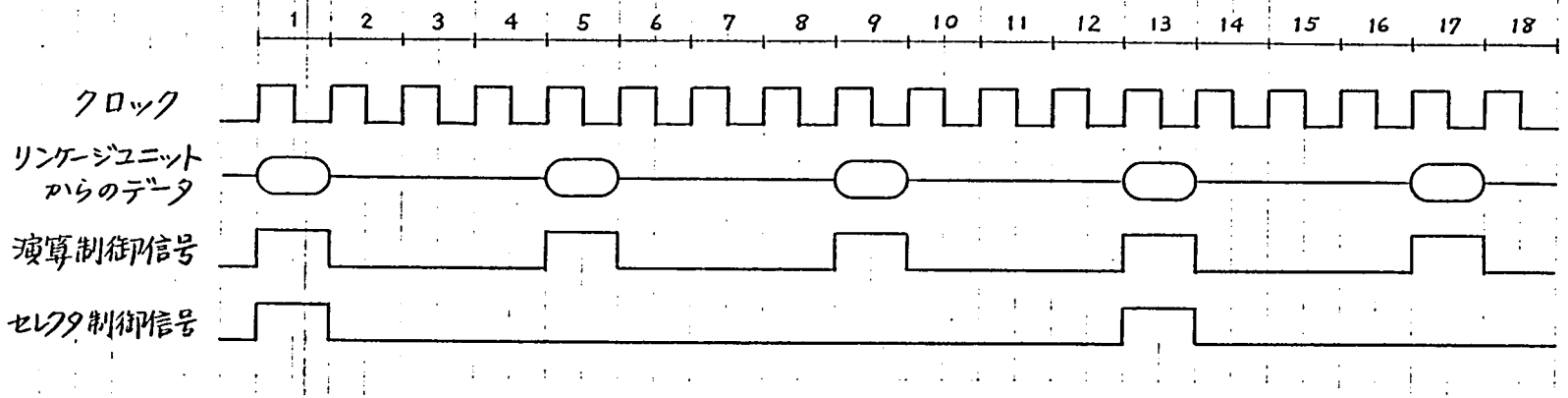


図４．６　エバリュエーションユニットのタイミングチャート

間のリンケージ演算の実行を考慮して、図４．７に示す構成を採っていて、図４．８に示す四つのモードで動作する。

図４．８(a)は、スティック長が１の場合で、図４．８(b)～(d)は、スティック長が２以上の場合である。スティック長が１の場合は、リンケージユニットの２入力ＡＬＵは、ＬＳＩ間のリンケージ演算専用なので、データのフィードバックはかからない。これに対してスティック長が２以上の場合は、図４．８(b)で、最初と２番目のＰＥ統合結果の再統合演算が実行され、図４．８(c)で、３番目以降のＰＥ統合結果が再統合される。そして、図４．８(d)では、ＬＳＩ間のリンケージ演算が実行され、図４．８(b)に戻って最終結果を出力する。この場合、図４．９に示すような制御信号が与えられなければならない。（ただし、図４．９は、スティック長が４の場合を示し、それぞれの制御信号が“１”の時、選択回路は、右側の入力データを選択するものとする。）

これら三つのユニットの動作タイミングを制御するため、ＩＳＰでは、ＬＳＩ外部から入力するタイミング制御信号を２ビットとし、それぞれの画素データの入力と同期させることにした。このタイミング制御信号は、同時に入力された画素データと同期してそれぞれのユニットに転送される。そして、シーケンスデータとしてタイミング制御に寄与する。つまり、それぞれのユニットでは、そのユニットの制御に必要なタイミング信号を、与えられたシーケンスデータから生成することになる。表４．２に、シーケンスデータの意味する内容を示す。なお、表４．２において、ＳＹＮＣ０およびＳＹＮＣ１は、シーケンスデータの外部端子名を示す。

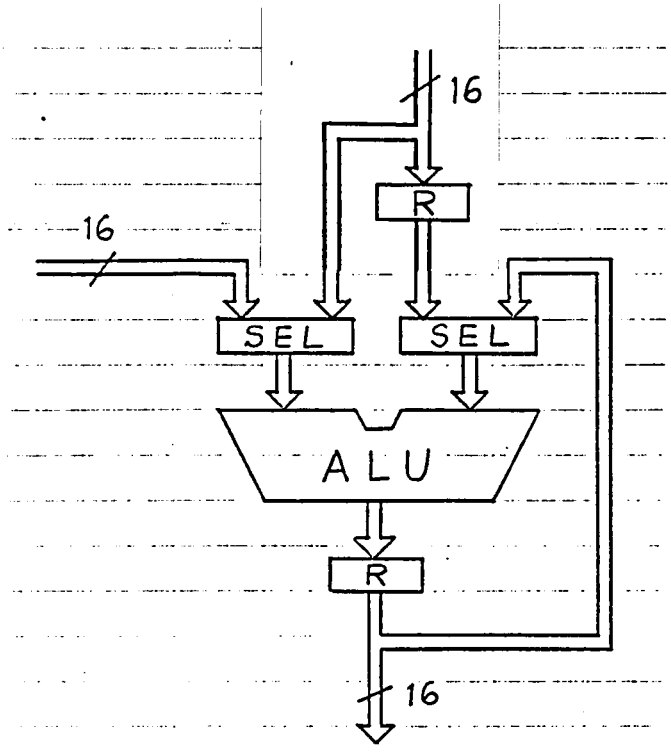


図４．７　リンケージユニットの２入力ＡＬＵの構成

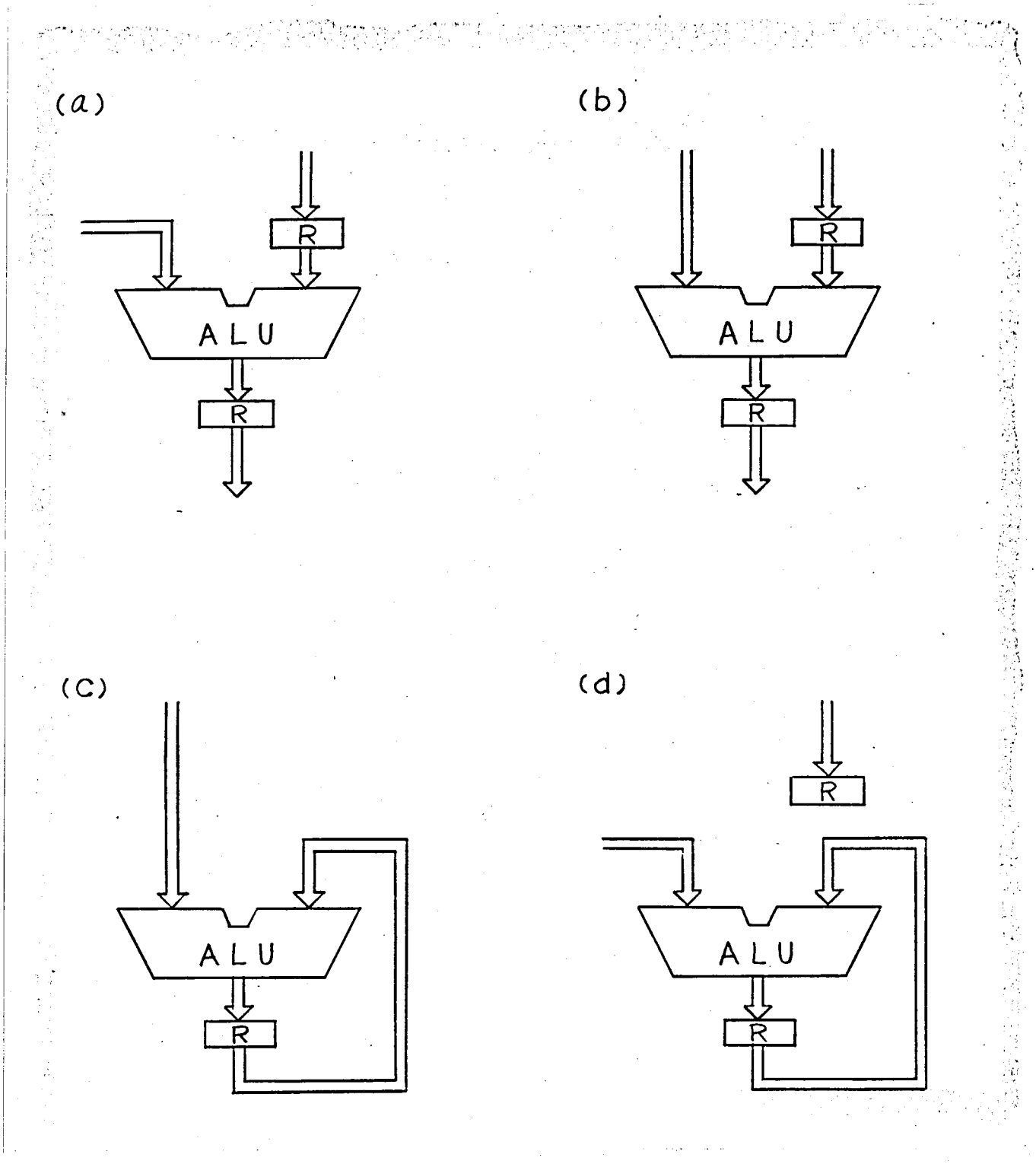


図４．８　リンケージユニットの２入力ＡＬＵの動作モード

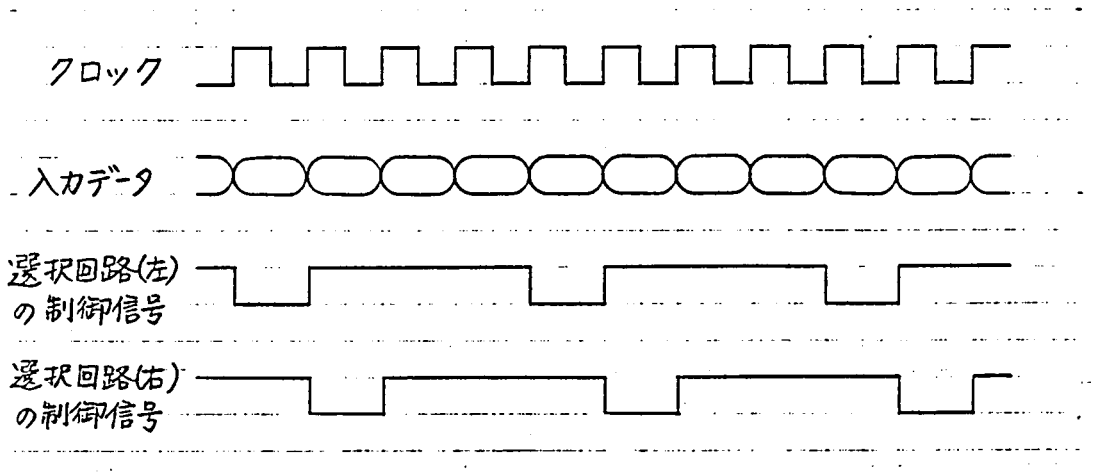


図４．９　リンケージユニットのタイミングチャート

表４．２　シーケンスデータの意味する内容

ＳＹＮＣ１	ＳＹＮＣ０	内　　容
1	1	スティックの先頭画素
1	0	再使用スティック＊の先頭画素 （マルチマスクオペレーション時のみ）
0	1	スティックもしくは再使用スティック＊の最終画素 （ただし、スティック長が２以上の時のみ）
0	0	上記以外の画素

＊　再使用スティックはＬＳＩ外部から入力するのではなく、内部で発生させるため、シーケンスデータのみ入力されて、ＬＳＩ内部で再使用スティックに付与される。

４．５　まとめ

空間積和演算やパターンマッチングなどのマスク演算では、２個以上のマスクデータを用いるマルチマスクオペレーションも、数多く提案されている。ここでは、前章までで明らかにされたＩＳＰのアーキテクチャにもとづいて、マルチマスクオペレーションを効率よく実行する方式について考察した。

画像処理システムを構築する観点からみると、オーバーヘッドを最小限に抑えることから、画像メモリの参照回数は少ない方が好ましい。そこで、ＩＳＰにおいては、カーネルを切り出す毎に、すべてのマスク演算を行なうと同時に、それらのマスク演算結果間の処理まで行なって、一つのカーネルに対する最終結果まで求めることにした。なぜなら、この方法によると、画像メモリの参照は、読み出しと書き込みの２回で済むからである。

ＩＳＰのアーキテクチャを考察する際、ＰＥへのデータ供給方式と、マスク演算結果間の処理方式の２点から検討した。また、ＰＥ増設方式とＰＥ節約方式の、二つのカーネル拡張方式のうち、ＰＥ節約方式に焦点をあてて検討した。なぜなら、ＰＥ増設方式に用いるラスタ走査は、ＰＥ節約方式に用いるスティック走査の特殊なものなので、ＰＥ節約方式で処理できれば、ＰＥ増設方式でも処理できるからである。

ＰＥへのデータ供給に関しては、マスクデータ単位の演算を、マスク数だけ繰り返す方式を採用した。この方式では、制御が容易である反面、同じ画像データを繰り返し入力しなければならない短所がある。この短所を補うために、データユニットのＳＲには、それぞれフィードバック機構を設けることにした。

マスク演算結果間の処理については、比較演算のみを対象とした。まず、エバリュエーションユニットに、二つの比較器と二つの閾値レジスタを設け、４種類の固定２値化処理を可能とした。次に、比較器の結果を用いて閾値レジスタの内容を書き換えられるようにし、マスク演算結果の最大値・最小値の摘出を可能とした。さらに、バイナリィカウンタを設けることにより、その最大値・最小値に関するマスク番号を生成できる。なお、比較演算と結果の出力を遅滞なく行なわせるため、初期値レジスタを設けた。

マルチマスクオペレーションのタイミング制御は、複数のＩＳＰを用いる場合を考慮して、外部から入力するシーケンスコードによる方式を採用した。シーケンスコードは２ビットとし、それぞれの画素データに付加して入力される。そして、データユニットのシフトレジスタのフィードバック機構や、リンケージユニットの２入力ＡＬＵのタイミング制御のほか、エバリュエーションユニットのバイナリィカウンタの初期化やカウントアップ、マスク演算結果間の処理結果の出力などを制御する。

４．６　第４章の参考文献

[1]Fukushima, T., Kobayashi, Y., Hirasawa, K., Bandoh, T., Ejiri, M. and Kuwahara, H. : An Image Signal Prosessor, IEEE ISSCC Digest of Technical Papers, Vol.26, pp.258～259 (1983).

[2]福島　忠、小林芳樹、平沢宏太郎、坂東忠秋、江尻正員、柏岡誠治、桑原　洋：画像処理用ＬＳＩ－ＩＳＰの開発、情報処理学会第２６回全国大会講演論文集、pp.939～940 (1983).

[3]福島　忠、小林芳樹、平沢宏太郎、坂東忠秋、江尻正員：画像処理用ＬＳＩ－Image Signal Processor のアーキテクチャ、電子通信学会論文誌(C)、Vol. J66-C, No. 12, pp.959～966 (1983).

[4]福島　忠、小林芳樹、平沢宏太郎、坂東忠秋、柏岡誠治、加藤　猛：多機能性を実現する画像処理用ＬＳＩ－ＩＳＰのアーキテクチャ、情報処理学会論文誌、Vol. 25, No. 5, pp.728～735 (1984).

[5]Prewitt, J. M. S. : Object Enhancement and Extraction, in Picture Processing and Psychopictorics, Academic Press, pp.75～149 (1970).

[6]Kirsch, R. : Computer Ddetermination of the Constituent Structure of Biological Images, Comput. Biomed. Res., Vol. 4, pp.315～328 (1971).

[7]Robinson, G. S. : Edge Detection by Compass Gradient Masks, CGIP, Vol. 6, pp.492～501 (1977).

[8]坂根茂幸、田村秀行：ＳＰＩＤＥＲ開発を通して観たディジタル画像処理アルゴリズムの現状(3)、情報処理学会コンピュータビジョン研究会資料４－４ (1980).

[9]柏岡誠治、江尻正員、坂本雄三郎：時分割パターン認識技術による群制御トランジスタ組立システム、電気学会論文集(C)、Vol. 96, No. 1, pp.9～16 (1976).

[10]福島　忠、小林芳樹、平沢宏太郎、坂東忠秋、柏岡誠治、加藤　猛：マルチマスクオペレーションを効率良く実行する画像処理用ＬＳＩ－ＩＳＰのアーキテクチャ、情報処理学会論文誌、Vol. 26, No. 2, pp.239～246 (1985).

[11]福島　忠、加藤　猛：画像処理用ＬＳＩ－ＩＳＰとその応用、O plus E、No. 5, pp.76～86 (1984).

[12]田村秀行（監修）：コンピュータ画像処理入門、総研出版 (1985).

[13]安居院猛、中嶋正之：コンピュータ画像処理、廣済堂産報出版 (1979).

第５章　画像処理システムの構築と性能評価

５．１　まえがき

さまざまな画像処理機能のうち、一般産業のいろいろな分野に実用化することを目的として、局所近傍演算に分類されるものを、高速に処理できる画像処理用ＬＳＩ－ＩＳＰを開発した[１]。実用化に際しては、応用対象毎に異なる要求を満たすため、ＩＳＰが備えるべき特徴として、カーネルの拡張方法[２]、各種の演算が実行できる多機能化[３]、さらにマルチマスクオペレーションへの対応方法[４]について、第２章、第３章および第４章で検討した。ここでは、これまで論じたＩＳＰのアーキテクチャを活用した、画像処理システムの構築方法について論ずる。

研究評価用の画像処理システムでは、高速処理よりは、むしろアルゴリズムの開発の上から、各種の画像処理機能の実行能力や、カーネルサイズの自由度の方が重要である。一方、オンライン用の画像処理システムでは、高速処理や小型軽量がより重要となってくる。また、各種の画像処理機能を実行できる多機能システムも、いくつかの単機能システムの最小公倍数的なシステムと見ることができる。そこで、２値・濃淡画像に対する基本演算を例にあげて、高速処理や小型軽量の観点から、ＩＳＰを活用した画像処理システムの構築方法を考察する。さらに、それらのシステムの処理性能について評価する。

５．２　画像処理システムの構築

まず、画像処理用ＬＳＩ－ＩＳＰを用いて、その特徴を活かした画像処理システムの構築方法について論ずる。任意の大きさのカーネルを扱える拡張性については、濃淡画像に対する空間積和演算と、２値画像に対するパターンマッチングを例にあげて、ＰＥ増設方式とＰＥ節約方式の、二つのカーネル拡張方式を考察する。各種の演算を実行する多機能性に対しては、濃淡画像における１次微分オペレータのいくつかについて述べる。また、マルチマスクオペレーションの例として、濃淡画像に対するPrewittのテンプレート型オペレータについて考察する。

５．２．１　ＰＥ増設方式によるシステムの構築

ＰＥ増設方式では、濃淡画像を処理する際、カーネルを構成する画素数と同数のＰＥを用いる。つまり、ＩＳＰ１個で１×４のカーネルを処理する。また、第２．３節で述べたように、入力画像はラスタ走査方式により走査される。これらのことを考慮して、カーネルが４×４の空間積和演算を実行するシステムは、図５．１に示すように構築できる[２]。

図５．１のシステムは、４個のＩＳＰと３個の遅延回路から構成されている。入力画像はラスタ走査方式により、ノンインタレーステレビ画像と同様に、左上隅から右下隅へ走査される。走査された画像データは、三つの遅延回路を介して、四つのＩＳＰに順次入力される。それぞれの遅延回路は、入力画像の１ラインを走査するのに要する時間だけ、画像データを遅延させるので、ＩＳＰへの４本の入力バスには、常に垂直方向に隣接した４個の画素データが、同時に取り出される。それぞれのＩＳＰのデータユニットには、遅延段数を変更できるＳＲが４個ずつ含まれているが、このＳＲの遅延段数をすべて１段とすると、四つのＩＳＰ内の１６個のＳＲには、４×４、合計１６個の隣接した画素データを切り出せる。あらかじめ、メモリユニットのＲＡＭに荷重係数を書き込んでおくと、ＳＲ

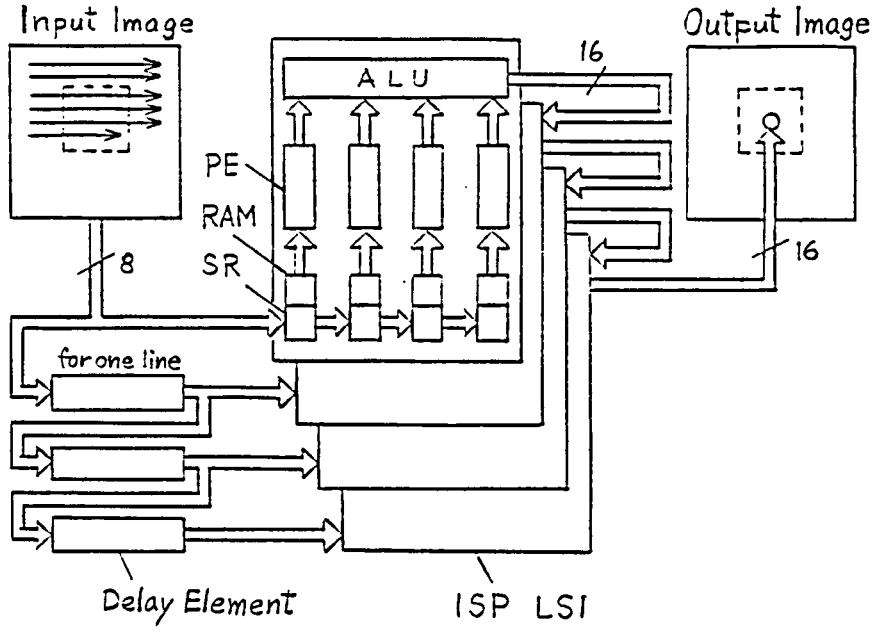


図５．１　ＰＥ増設方式により構築した４×４
空間積和演算を実行するシステム

内の画素データは、ＲＡＭ内の荷重係数と、ＳＲの対応するＰＥで掛け合わされて、リンケージユニットのＡＬＵで順次加算される。ＬＳＩ間リンケージ演算にともなうオペランド間の時間歪みは、第２章で述べたように、画像データの入力バッファとしての歪み補正バッファにより補正される。最終結果は出力画像の画素データとして、画像メモリに書き込まれる。４×４のカーネルは、画像の走査につれて左上隅から右下隅へ移動するため、入力画像の全域にわたって、４×４空間積和演算が可能となる。

図５．１では、ＩＳＰを垂直方向に増設してカーネルを拡張しているが、垂直方向だけでなく水平方向にもＩＳＰを増設して、２次元的にカーネルを拡張できる。図５．２は、カーネルを２次元的に拡張して、８×８のカーネルに対する空間積和演算を実行するためのシステム構成である[５,６]。

図５．２のシステムは、１６個のＩＳＰと７個の遅延回路から構成されている。図５．１のシステムと同様に、７個の遅延回路は、入力画像の１ラインを走査するのに要する時間だけ、画像データを遅延させる。また、左列のＩＳＰのＳＲを介した画像データは、右列のＩＳＰのＳＲに与えられる。このため、１６個のＩＳＰ内の６４個のＳＲには、８×８のカーネルが切り出される。それぞれのＳＲに切り出された画素データは、対応するＰＥで、ＲＡＭからの荷重係数と掛け合わされ、後段のＡＬＵで加算される。この時、６４個のＰＥが並列処理することになる。

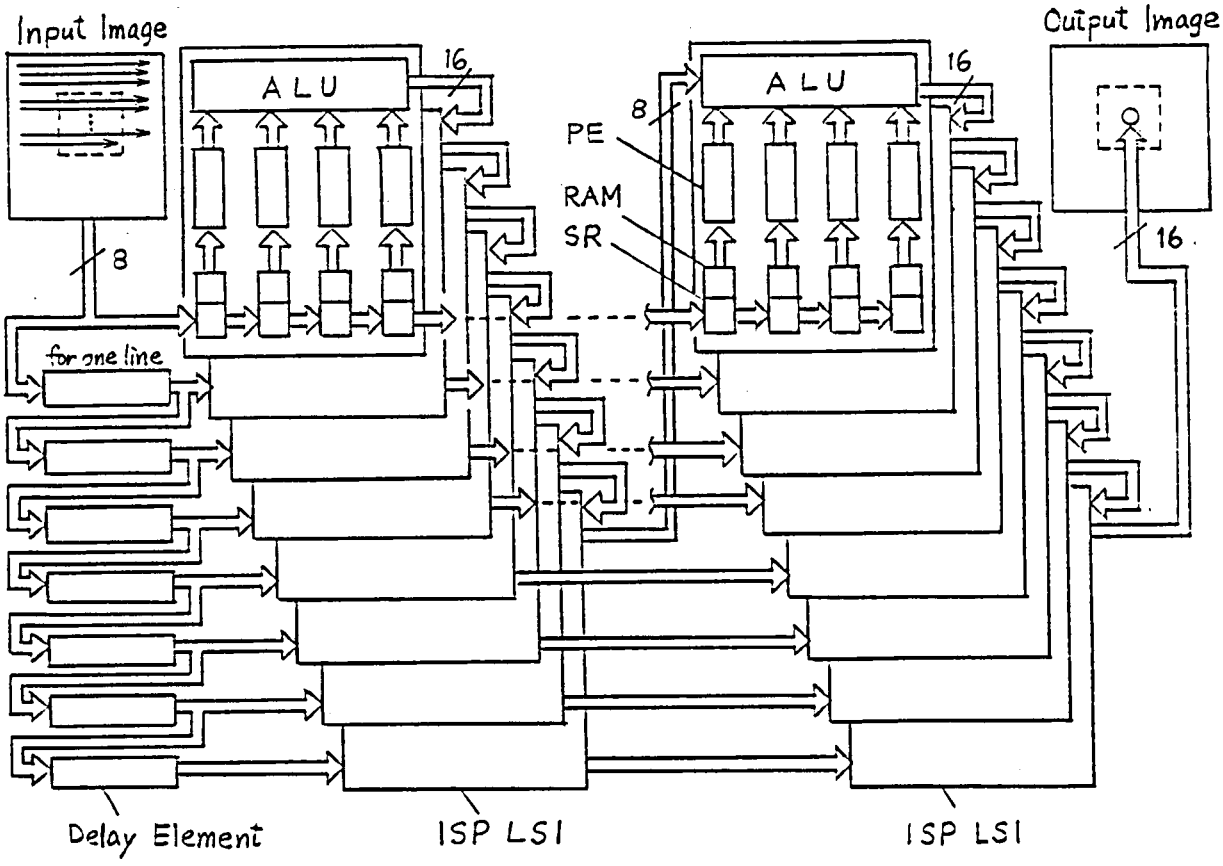


図５．２　ＰＥ増設方式により構築した８×８
空間積和演算を実行するシステム

図５．１および図５．２において、画像データの歪み補正バッファは図示されていないが、ｉ番目にリンケージされたＩＳＰの歪み補正バッファの、遅延段数Ｄｉを求める一般式は、次の手順で求められる。

まず、図５．１のように、ＩＳＰが１次元的に配列された場合を考える。歪み補正バッファは、画像データの入力バッファでもあるため、リンケージされた最初のＩＳＰの、歪み補正バッファの遅延段数Ｄ1は、常に１となる。リンケージ演算におけるパイプライン段数は、第２．４節に示すように、ＩＳＰ１個につき２段である。つまり、ｉ番目のＩＳＰにおける歪み補正バッファの遅延段数Ｄｉは、

$$Di = 2i - 1$$

と表わすことができる。つまり、図５．１に示すＩＳＰの歪み補正バッファの遅延段数は、上からそれぞれ１、３、５、７段である。

次に、図５．２のように、ＩＳＰを２次元的に配列する場合を考える。この場合、遅延回路から直接、画像データが供給される左列のＩＳＰについては、上述の場合と全く同じであるが、左列のＩＳＰを介して画像データの供給を受ける右列のＩＳＰについては、画像データを供給する左列のＩＳＰの、歪み補正バッファの遅延段数Ｄｊを継承することになる。つまり、リンケージがｉ番目のＩＳＰにおける歪み補正バッファの遅延段数Ｄｉは、

$$Di + Dj = 2i - 1$$
$$Di = 2i - 1 - Dj$$

となる。より一般的には、画像データが複数個のＩＳＰを経て、注目のＩＳＰに供給される場合は、通過するＩＳＰの歪み補正バッファの遅延段数すべてを継承することになる。そのため、注目のＩＳＰの歪み補正バッファの遅延段数Ｄｉは、

$$Di + \sum_{j} Dj = 2i - 1$$
$$\therefore \quad Di = 2i - 1 - \sum_{j} Dj$$

と一般化することができる。つまり図５．２において、左列のＩＳＰの歪み補正バッファの遅延段数は、上からそれぞれ１、３、５、７、９、11、13、15段であり、右列のそれらはすべて16段となる。

ＩＳＰの歪み補正バッファの遅延段数は、チップ面積の問題から最大１６段としたため、８×８のカーネルの演算では、図５．２に示すように、歪み補正バッファにより、リンケージ演算のオペランド間の歪み補正が可能である。より一般的に言うと、垂直方向にＩＳＰをリンケージする際、８個までならば歪み補正が可能である。つまり、縦８×横ｎのカーネルに対しては、歪み補正バッファだけで歪み補正できる。

次に、パターンマッチングを例にあげて、２値画像を処理するシステムの構築法を考察

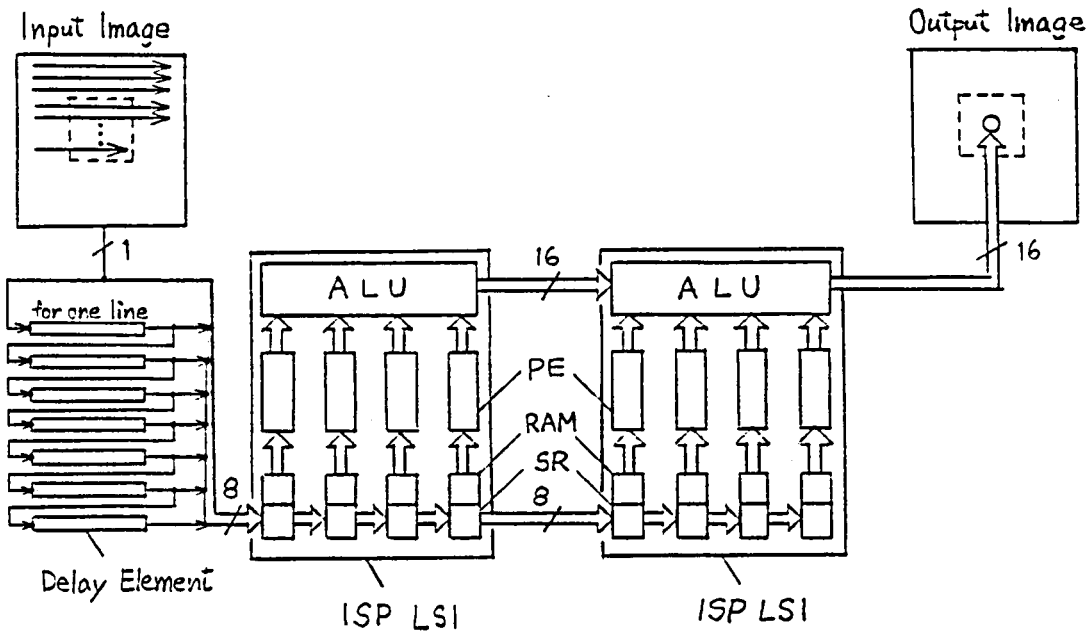


図５．３　ＰＥ増設方式により構築した８×８パターン
マッチングを実行するシステム

する。

ＰＥ増設方式で２値画像のパターンマッチングを処理する場合は、一つのＰＥは８個の画像データのマッチングを計算する。つまり、１個のＩＳＰでは、８×４画素のパターンマッチングが可能である。図５．３および図５．４に、それぞれカーネルが８×８、１６×１６のパターンマッチングを実行するシステムの構成を示す。

図５．３のシステムは、２個のＩＳＰと７個の遅延回路から構成されている。図５．３では２値画像を扱うため、遅延回路のビット幅は１である。７個の遅延回路により、垂直方向に隣接した８個の画素データが、左側のＩＳＰに与えられる。また、図５．４のシステムは、８個のＩＳＰと、ビット幅１の遅延回路１５個から構成されている。それぞれのシステムにおいて、ＳＲに切り出された画素データは、ＰＥにおいてＲＡＭから読み出されたテンプレートとの一致画素数が算出され、リンケージユニットのＡＬＵで、一致画素数の総和が算出される。なお、それぞれの歪み補正バッファの遅延段数は、空間積和演算の場合と同様に、ＩＳＰの配置とリンケージ順序により求められる。つまり、図５．３では、左側のＩＳＰの歪み補正バッファの遅延段数は１段で、右側のそれは２段である。また、図５．４においては、左列の左端上下のＩＳＰのそれはそれぞれ１、３段であり、残りのＩＳＰのそれは、すべて４段である。

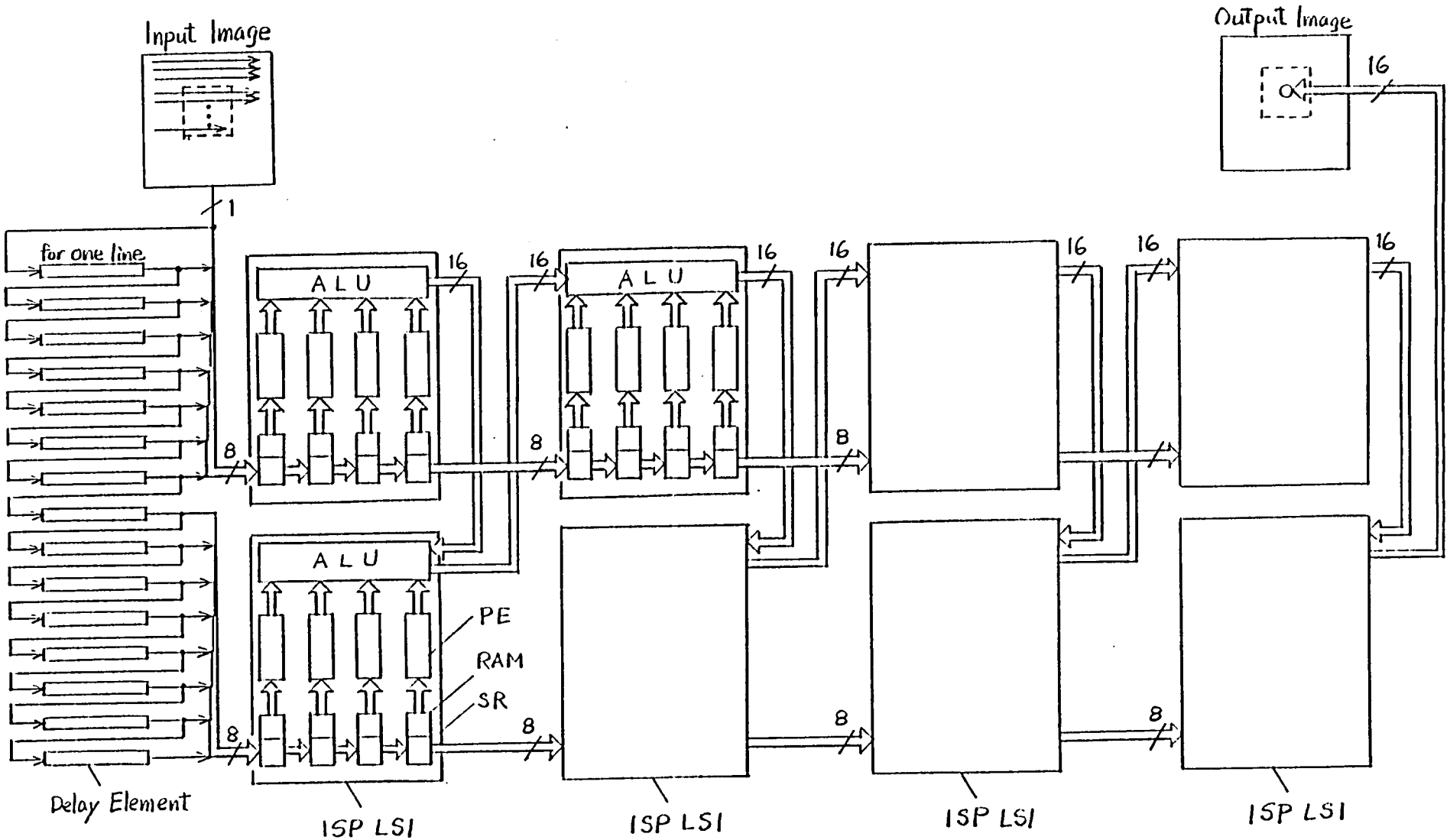


図５．４　ＰＥ増設方式により構築した16×16パターン
マッチングを実行するシステム

５．２．２　ＰＥ節約方式によるシステムの構築

ＰＥ節約方式では、濃淡画像を処理する際、カーネルを形成する画素数より少ないＰＥ数を用いる。つまり、ＩＳＰ１個で最大４×４のカーネルを処理する。また、第２．３節で述べたように、入力画像の走査にはスティック走査を用いる。これらのことから、４×４空間積和演算を実行するシステムを、ＰＥ節約方式で構築すると、図５．５に示すようなシステムとなる[２]。

図５．５のシステムは、１個のＩＳＰのみから構築されている。入力画像はスティック長が４のスティック走査により走査される。ＩＳＰ内の４個のＳＲは、それぞれ４段の遅延回路として、カーネルを形成する４×４の画素データを切り出す。切り出された画素データは、４回に分けて、ＲＡＭ内の荷重係数とＰＥで掛け合わされ、後段のＡＬＵで足し合わされ、出力画素データとなる。つまり、１個の出力画素を算出するのに４マシンサイクル要することになる。ＰＥ増設方式で構築した図５．１のシステムと比較すると、４倍の処理時間を必要とする一方、ＩＳＰは 1/4 の１個でよい上に、カーネル切り出しのための周辺回路を必要としない。なお、歪み補正バッファは、遅延段数１段で入力バッファとしてのみ用いられる。

ＰＥ節約方式では、ＩＳＰ１個で４×４までのカーネルしか処理できないので、カーネルが８×８の空間積和演算を処理する場合、４個のＩＳＰが必要である。図５．６は、

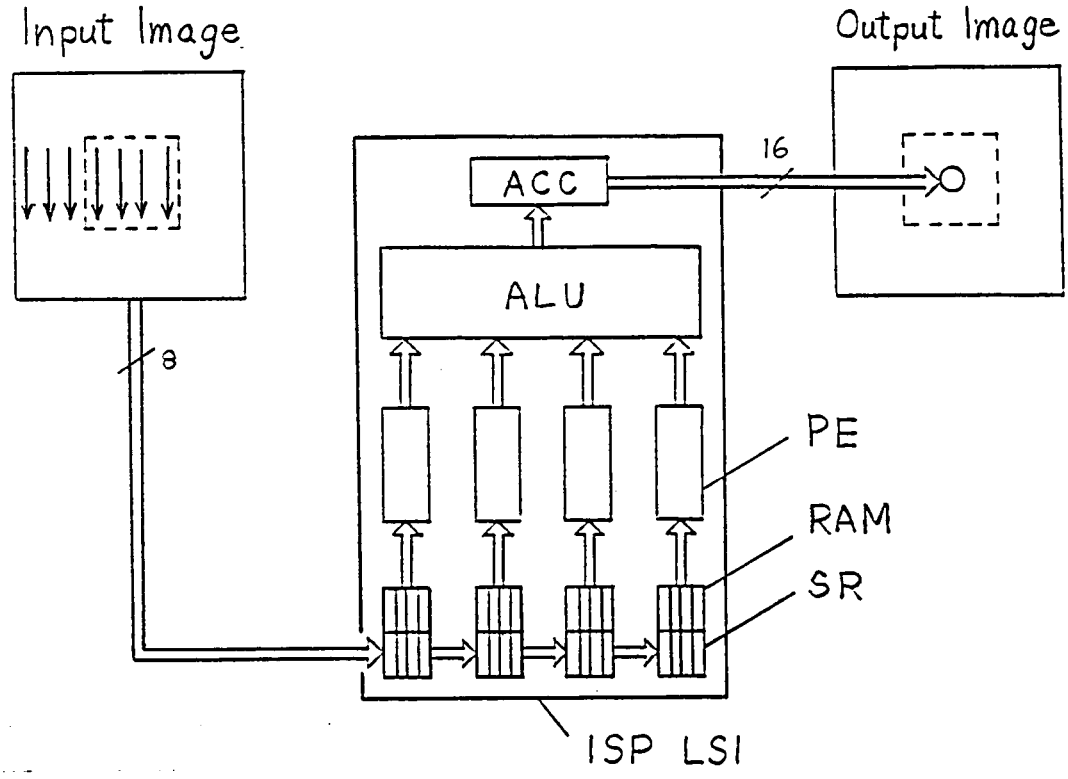


図５．５　ＰＥ節約方式において構築した４×４
空間積和演算を実行するシステム

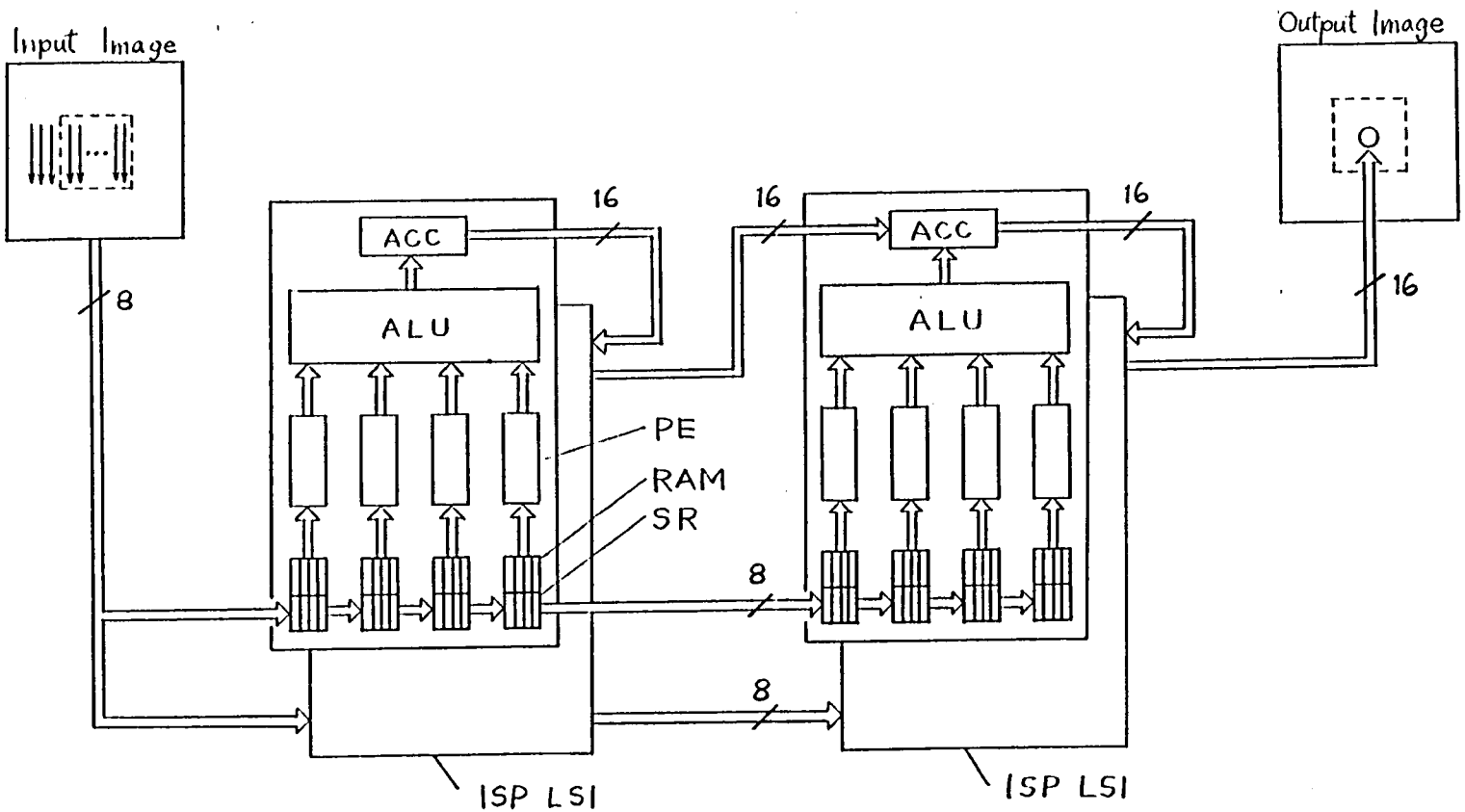


図５．６　ＰＥ節約方式において構築した４×４
空間積和演算を実行するシステム

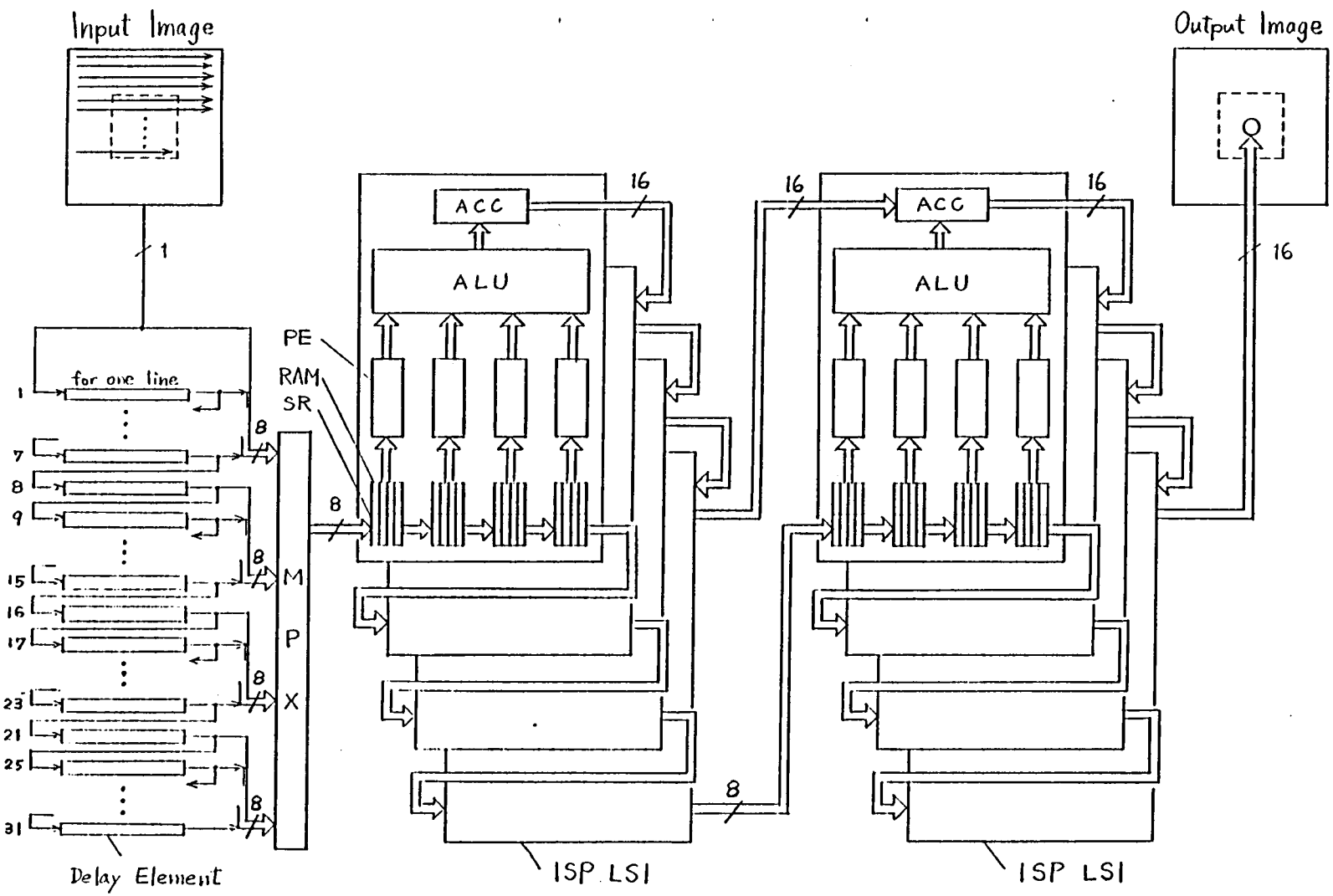


図５．７　ＰＥ節約方式において構築した32×32パターン
マッチングを実行するシステム

ＩＳＰを４個用いて、８×８空間積和演算を実行するシステムの構成を示す[５,６]。入力画像は、スティック長が８のスティック走査で走査される。走査された画素データは、左列の二つのＩＳＰに、交互に入力される。ＩＳＰ内のＳＲの遅延段数はすべて４段で、右列のＩＳＰには、左列のＩＳＰのＳＲを介して、画素データが供給される。４個のＩＳＰ内の１６個のＳＲに切り出された８×８個の画素データは、４回に分けて、それぞれのＲＡＭにあらかじめ記憶されてあった荷重係数と掛け合わされ、４個の部分積和にまとめられる。そしてＩＳＰ間のリンケージ演算により、出力画素データが求められる。この時の歪み補正バッファの遅延段数は、ＰＥ増設方式と同じように、ＩＳＰの配置とリンケージ順序により求められる。

図５．６のシステムにおいて、ＩＳＰを効率よく動作させるには、ＩＳＰの動作速度の２倍の速度で画像を走査しなければならない。この条件を満たす限りは、ＰＥ増設方式による図５．２のシステムと比べ、1/4 の処理速度ながら、1/4 のＩＳＰ数の上、カーネル切り出しのための周辺回路なしで、８×８空間積和演算を実行できる。

ＰＥ節約方式で２値画像のパターンマッチングを処理する場合は、１個のＩＳＰで最大３２×４個の画素データを処理できる。前項でも述べたように、パターンマッチングを処理する場合は、一つのＰＥで一度に処理できる画素データの数は８ビットの８個にすぎない。そのため、ＰＥ増設方式においては、垂直方向に隣接する８個の画素データをそれぞれのＰＥに与え、１個のＰＥで８×４画素のカーネルを処理している。ＰＥ節約方式は、ＰＥを時分割に使用して、同時に処理できる画素数よりも大きなカーネルを処理する方式である。その時分割数が最大４であるため、１個のＩＳＰでは、８×４×４個の画素から成るカーネルの処理が可能である。ところが、カーネルは処理を終えると１画素ずつ移行してゆかなければならない。そのため、一つのＰＥで時分割に処理される画素データは、すべて垂直方向に隣接していなければならない。つまり、ＰＥ節約方式において、ＩＳＰ１個が処理できる２値画像のカーネルは、最大　３２×４画素となる。そこで、ＩＳＰを８個用いると、カーネルを３２×３２画素まで拡張できる。図５．７に、カーネルが３２×３２のパターンマッチングを実行する、画像処理システムの構成を示す。

図５．７のシステムは、８個のＩＳＰと３１個の遅延回路、および4-to-１の８ビットセレクタ(selector or multiplexer)から構成される。入力画像はラスタ走査により走査される。走査された画素データは、遅延回路に入力される。それぞれの遅延回路は、入力画像の１ラインを走査する時間だけ、画素データを遅延させるので、垂直方向に隣接する３２個の画素データが取り出される。３２個の画素データ、３２ビットは、８ビットずつ四つに分割され、セレクタで順次選択され、左端のＩＳＰに入力される。ＩＳＰ内のＳＲの遅延段数はそれぞれ４段で、８×４ビットの画素データを一時記憶し、時分割数が４で動作するＰＥは、それぞれ垂直方向に隣接する３２個の２値画素データを処理する。つまり、１個のＰＥが３２×１のカーネルを処理するため、３２個のＰＥで３２×３２のカーネルが処理でき、４マシンサイクル毎にカーネルを、右方向へ１画素ずつ移行することが可能となる。

図５．７のシステムにおいて、入力画像はラスタ走査により走査されるが、遅延回路により取り出される３２ビットの画素データを、８ビットずつ４個の濃淡画素データとみな

すと、スティック長が４のスティック走査により走査されたデータが、ＩＳＰに入力されると見ることができる。つまり、３１個の遅延回路と４-to-１セレクタは、ラスタ走査をスティック走査へ変換する回路である。なお、図５．７のシステムでは、画像の走査速度はＩＳＰの動作速度の 1/4 である。

図５．７に習って、濃淡画像におけるラスタ走査も、任意のスティック長のスティック走査に変換できる。図５．８は、濃淡画像におけるラスタ走査を、スティック長が４のスティック走査に変換する回路である。この場合も図５．７におけると同様に、画像の走査速度はＩＳＰの動作速度の 1/4 である。一般に、ラスタ走査をスティック走査に変換する場合、画像の走査速度はＩＳＰの動作速度の、1/(スティック長) になる。

５．２．３　１次微分オペレータを実行するシステムの構築

表３．１に示したように、これまで各種の１次微分オペレータが提案されているが、ここでは、表３．１の(３)および(６)を例にとって[７,８]、ＩＳＰによるシステムの構築について述べる。

まず。表３．１の(３)は、２×２個の入力画素データを、ａ、ｂ、ｃ、ｄ、出力画素データをｘとすると、

$$x = [\,|a-b+c-d|+|a+b-c-d|\,]/2.$$

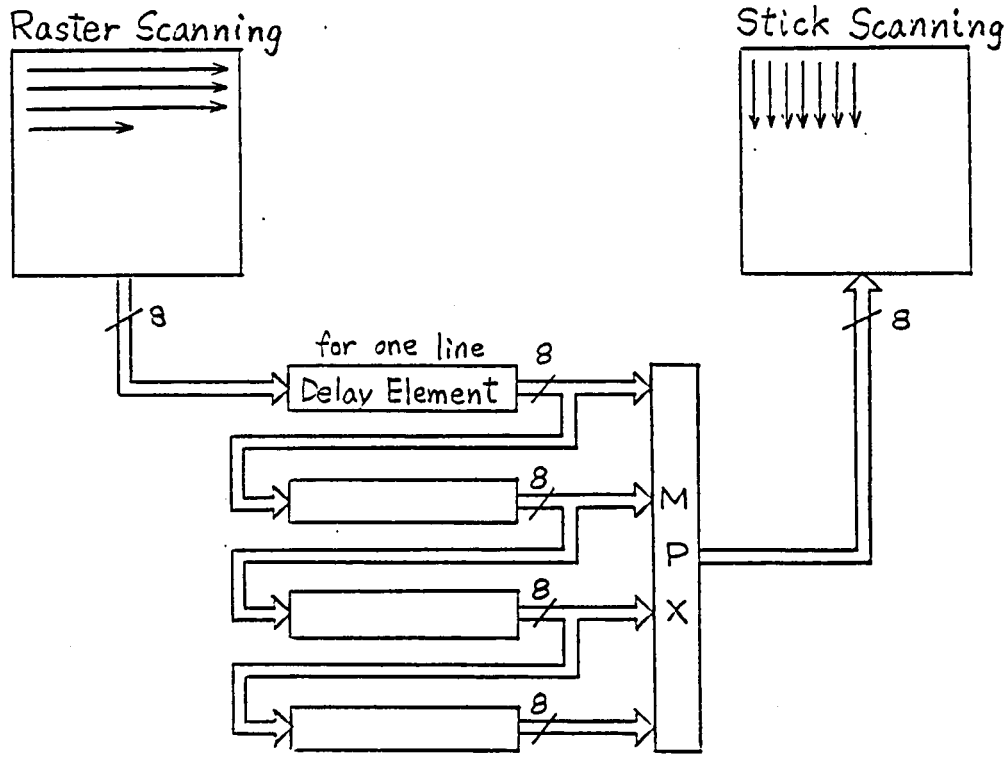


図５．８　ラスタ走査をスティック長４の
スティック走査に変換する回路

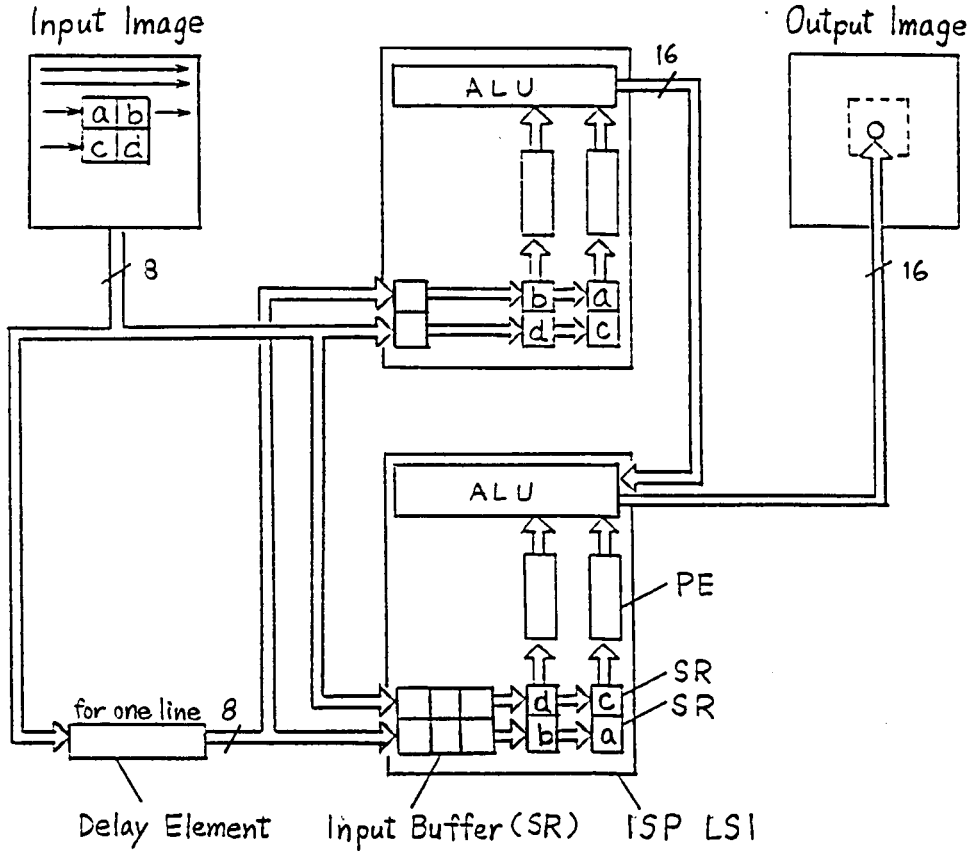


図５．９　１次微分オペレータを実行するシステム(1)

と表わせる。これが次のように書き直せる。

$$x = |(a+c)-(b+d)|/2 + |(a-c)+(b-d)|/2.$$

この式で表わされる１次微分オペレータは、図５．９に示すシステムで実行される。

図５．９のシステムは、ＰＥ増設方式により、２個のＩＳＰと１個の遅延回路から構成されている。入力画像はラスタ走査で走査され、８個あるＰＥのうち、４個のＰＥのみ使用される。図５．９に示すように、走査された画素データは、直接ＩＳＰのＡバスに入力されるとともに、遅延回路を介してＩＳＰのＢバスにも与えられる。ＩＳＰに入力された画素データは、歪み補正バッファを介して、１段の遅延段数をもつＳＲに一時記憶される。つまり、それぞれのＩＳＰのＳＲには、歪み補正バッファの段数の違いから時間ずれはあっても、同じ２×２のカーネルを成す４個の画素データが取り出されることになる。前段のＩＳＰで $|(a+c)-(b+d)|/2$ を、後段のＩＳＰで $|(a-c)+(b+d)|/2$ と前段のＩＳＰの結果との総和、を算出することにより、出力画素データを求めることができる。

また、表３．１の(6)は、図５．１０に示すシステムにより実行できる。図５．１０のシステムは、２個のＩＳＰと２個の遅延回路から構成されており、入力画像はラスタ走査により走査される。走査された画素データは、直接、および２個の遅延回路を介して、それぞれのＩＳＰのＡバスに入力され、１個の遅延回路を介した画素データは、それぞれの

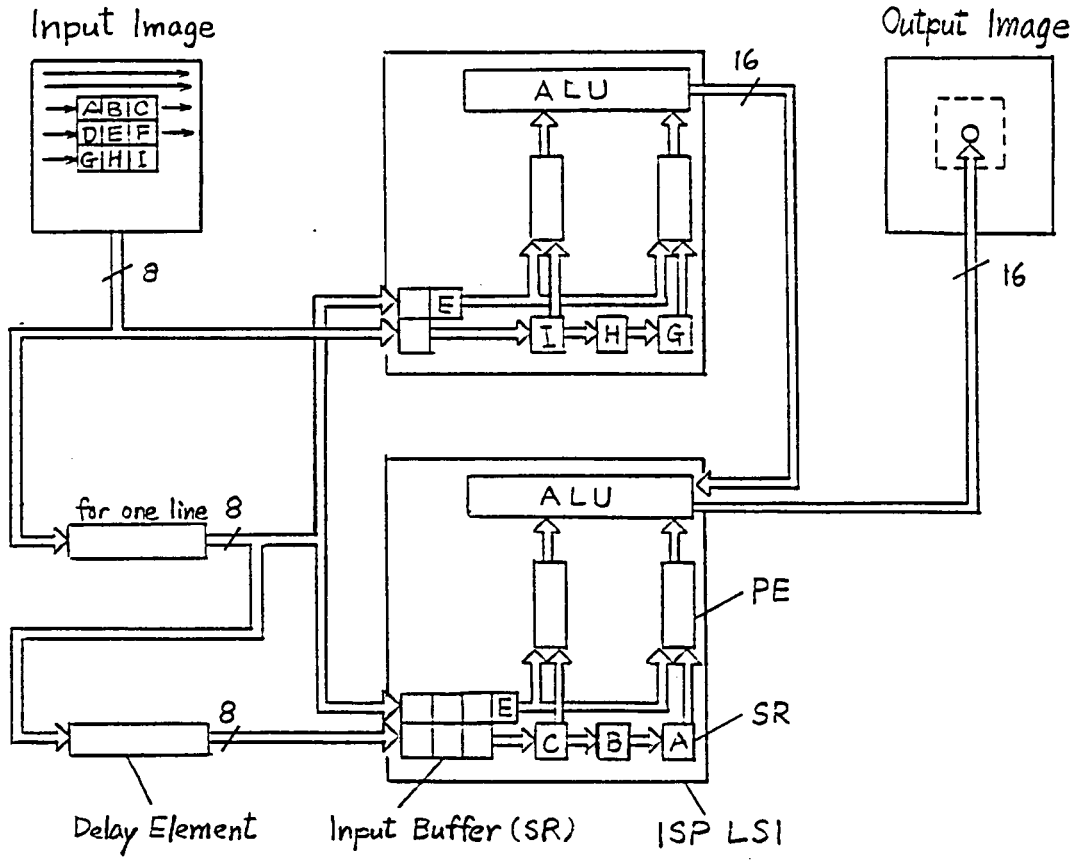


図５．１０　１次微分オペレータを実行するシステム(２)

ＩＳＰのＢバスに与えられる。いずれのＩＳＰにおいても、Ｂバスの歪み補正バッファは、Ａバスのそれより遅延が１段多いため、Ａバスを介して４個のＰＥに、画素データＡ、Ｃ、Ｇ、Ｉ、が与えられる時、同時に、３×３のカーネルの中央の画素データＥが与えられることになる。つまり、前段のＩＳＰでは |Ｇ－Ｅ|＋|Ｉ－Ｅ| を、後段のＩＳＰでは |Ａ－Ｅ|＋|Ｃ－Ｅ| と前段のＩＳＰとの総和、を算出することにより、出力画素データを求めることができる。

５．２．４　マルチマスクオペレーションを実行するシステムの構築

マルチマスクオペレーションを実行するシステムの構築については、表４．１に示したものの中から、Ｐrewittのテンプレート型オペレータ[９]を例にあげて論ずる[４,10]。

Ｐrewittのテンプレート型オペレータは、カーネルが３×３の空間積和演算を実行することが基礎となる。通常の３×３空間積和演算を実行するシステムは、ＰＥ増設方式によってもＰＥ節約方式によっても構築できる。しかし、マルチマスクオペレーションを実行する場合、マスクデータが複数個になることから、メモリユニットのＲＡＭ容量を考慮しなければならない。ＩＳＰにおいては、それぞれのＰＥに対して１６バイトのＲＡＭ容量が割り当てられている。Ｐrewittのテンプレート型オペレータは、３×３のマスクデータを８個用いる。そのため、ＰＥ増設方式でシステムを構築する場合は、それぞれのＰＥが、一つのカーネル当たり１個の画素データを処理することから、ＰＥ一つ当たり８バイトのＲＡＭ容量があればよい。しかし、ＰＥ節約方式でシステムを構築すると、それぞれのＰＥが、一つのカーネル当たり３個の画素データを処理することから、ＰＥ一つ当たり２４バイトのＲＡＭ容量が必要になる。ＩＳＰでは、一つのＰＥに１６バイトのＲＡＭ容量しか用意されていないため、ＰＥ増設方式ならばシングルシステムで処理できても、ＰＥ節約方式ではデュアルシステムでなければ処理できなくなる。Ｐrewittのテンプレート型オペレータを実行するシステムにつき、ＰＥ増設方式およびＰＥ節約方式により構築したものを、それぞれ図５．１１と図５．１２に示す。

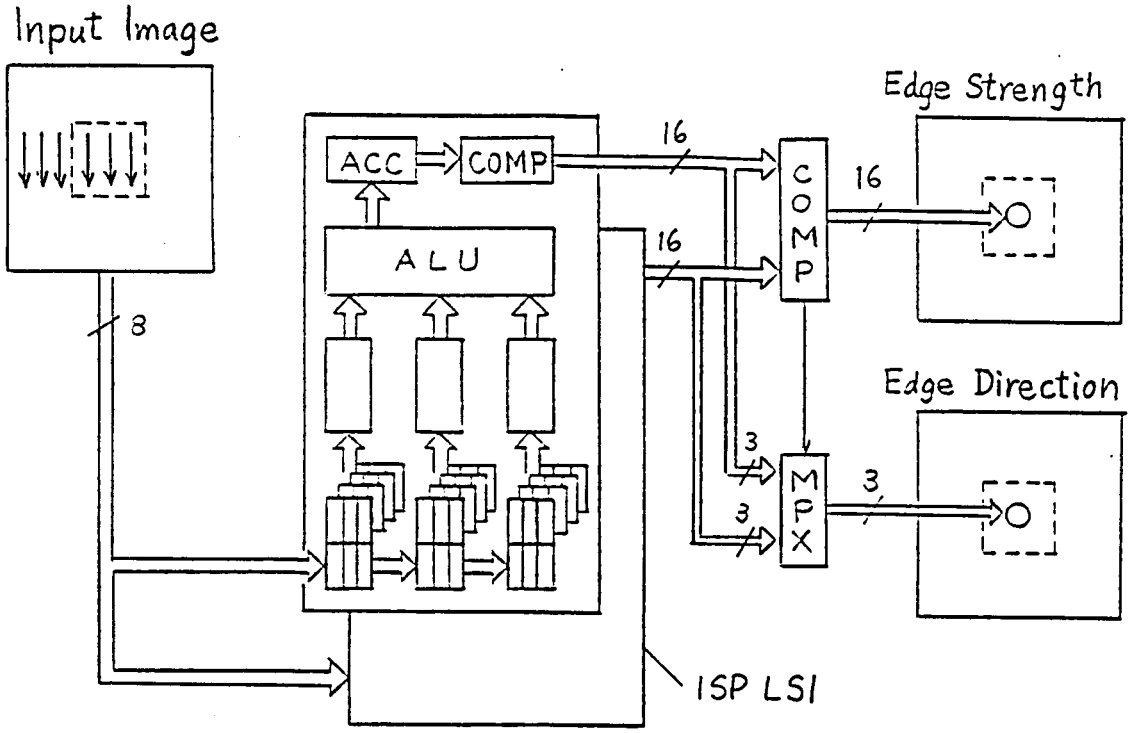


図５．１２　ＰＥ節約方式において構築したＰrewittの
テンプレート型オペレータを実行するシステム

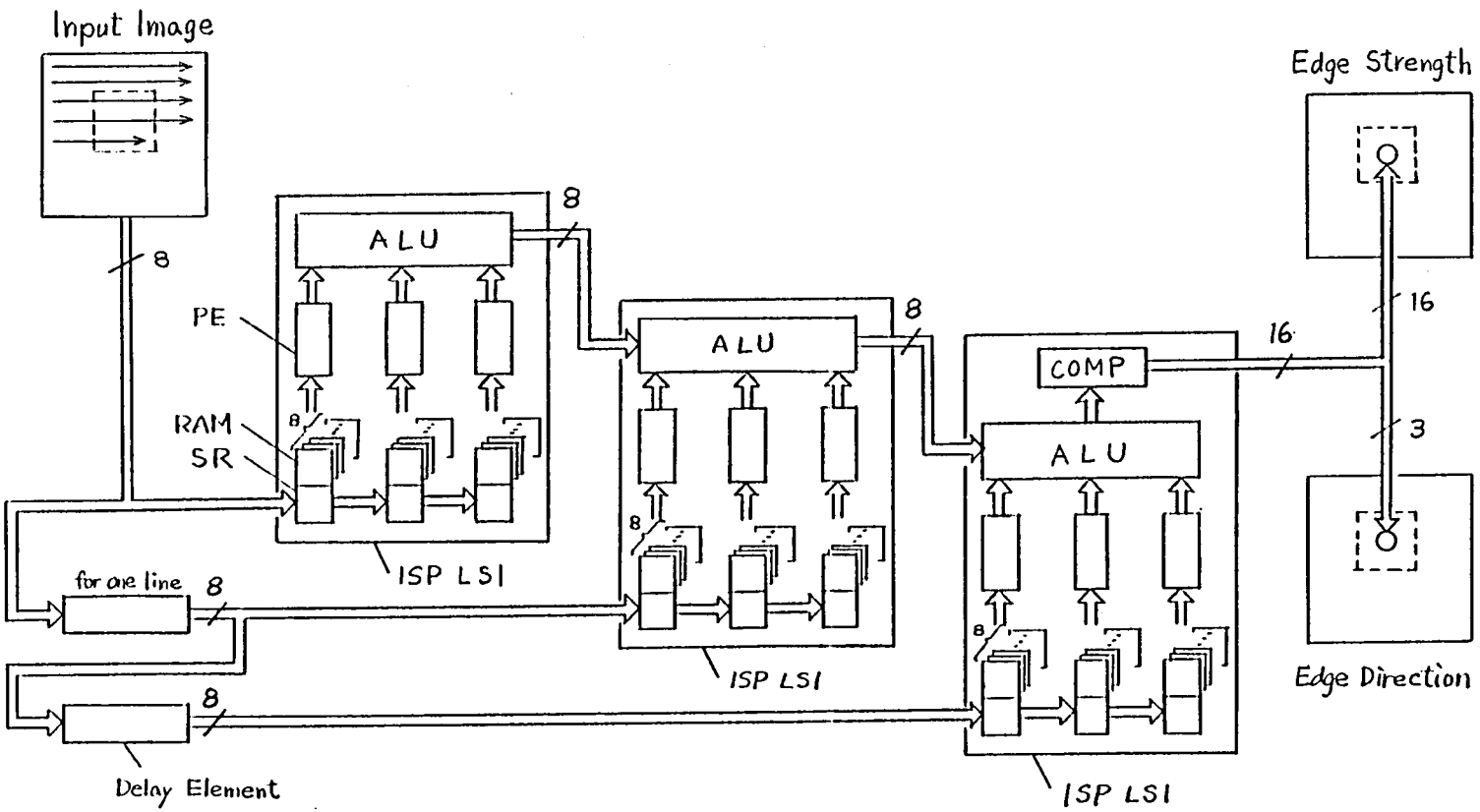


図５．１１　ＰＥ増設方式において構築したPrewittの
テンプレート型オペレータを実行するシステム

図５．１１のシステムにおいて、入力画像はラスタ走査により走査される。画素データは８マシンサイクルに一度ずつ走査される。走査された画素データと垂直に隣接する２個の画素データ、合計３個の画素データが、８マシンサイクルに一度ずつＩＳＰに入力される。３個のＩＳＰは、１マシンサイクル毎に１個の３×３マスクデータとの空間積和演算を実行し、８マシンサイクルにわたって、８個のマスクデータとの積和値を求める。求められた８個の積和値は、エバリュエーションユニットで比較され、その最大値と最大値を生んだマスク番号とを、結果として最終のＩＳＰのリンケージバスから出力する。

図５．１２のシステムにおいては、それぞれのＩＳＰが、一つのカーネル当たり、４個の３×３マスクデータを処理することになる。つまり、マスク数４のマルチマスクオペレーションを２組同時に行なって、それぞれの最大演算結果を比較して、最終の結果を求めるのである。入力画像は、スティック長が３のスティック走査で走査される。それぞれのＩＳＰがマスク数４のマルチマスクオペレーションを実行するので、３個の画素データから成る一つのスティックは、１２マシンサイクル毎に走査されて、同時に二つのＩＳＰに入力される。それぞれのＩＳＰは３マシンサイクル毎に、１個の３×３マスクデータとの空間積和演算を実行し、１２マシンサイクルにわたって、それぞれ４個ずつマスクデータとの積和値を算出する。そして、それぞれのＩＳＰで４個の積和値の比較演算の後、最大値と最大値のマスク番号を出力することになる。そこで、それぞれのＩＳＰからの最大値結果を比較して、最終結果としての最大積和値とそのマスク番号を決定する回路が必要となる。

図５．１１および図５．１２のシステムにおいて、画素データとともに２ビットのシーケンスデータが、ＩＳＰに入力される。シーケンスデータは、画素データと同じように、歪み補正バッファやＳＲで遅延させられ、データユニット、リンケージユニット、エバリュエーションユニットの三つのユニットで、同時に入力された画素データの演算の制御に寄与する。ラスタ走査はスティック走査の特殊なものーースティック長が１のスティック走査ーーであるから、入力画像がスティック走査で走査される場合を例にとると、２ビットのシーケンスデータは、ＩＳＰに入力される画素データが、表４．２に表わされる内容を示すよう、ＩＳＰに入力される。つまり、図５．１１および図５．１２に示すシステムにおける入力データのタイミングチャートは、それぞれ図５．１３、図５．１４のように表わせる。図５．１３と図５．１４において、ＣＬＫはクロック信号、Ａは画素データの入力信号、ＳＹＮＣはシーケンスデータを示す。

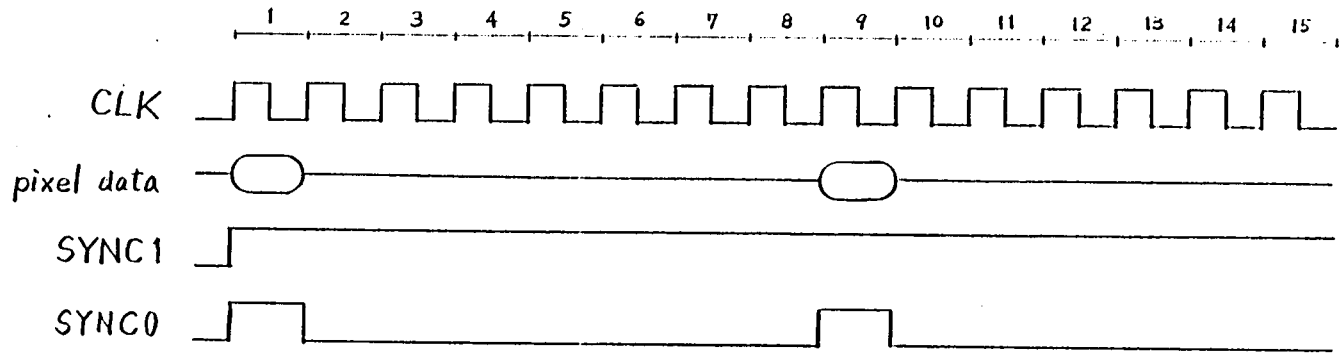


図５．１３　図5.11に示すシステムにおける入力データのタイミングチャート

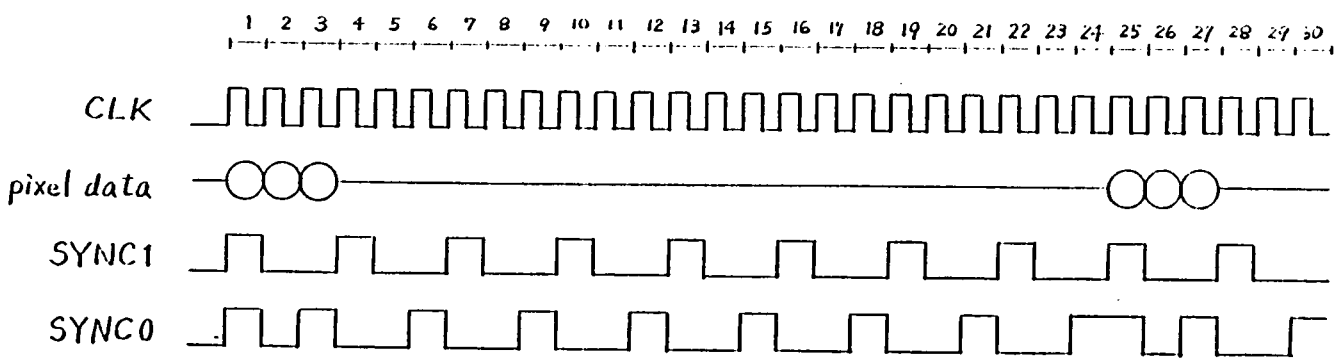


図５．１４　図5.12に示すシステムにおける入力データのタイミングチャート

５．３　画像処理システムの性能評価

前節で論じた画像処理システムの処理性能を、表５．１に示す。表５．１には、それぞれの機能について、用いたＩＳＰの個数、ＩＳＰを６ＭＨｚで動作させた時のＭＯＰＳ(Million Operations Per Second)値、２５６×２５６画素および５１２×５１２画素から成る画像を処理する時間を示している。

ノンインタレース（非飛び越し）走査のテレビ画像は、１秒間に６０枚生成される。このテレビ画像を実時間で処理するためには、一画面を１６．７ｍｓで処理しなければならない。表５．２から分かるように、図５．１、図５．２、図５．３、図５．４、図５．９、および図５．１０のシステムは、２５６×２５６画素サイズの画像ならば、実時間処理が可能である。

表５．１　画像処理システムの処理機能

図	機　能	ＩＳＰ数	ＭＯＰＳ	処　理　時　間	
				256×256画素	512×512画素
５．１	４×４空間積和演算	４	１８６	１０．９ms	４３．７ms
５．２	８×８空間積和演算	１６	７６２	１０．９ms	４３．７ms
５．３	８×８パターンマッチング	２	７６２	１０．９ms	４３．７ms
５．４	16×16パターンマッチング	８	３０６６	１０．９ms	４３．７ms
５．５	４×４空間積和演算	１	４６．５	４３．７ms	１７４．８ms
５．６	８×８空間積和演算	４	１９０．５	４３．７ms	１７４．８ms
５．７	32×32パターンマッチング	８	３０７０．５	４３．７ms	１７４．８ms
５．９	１次微分オペレータ	２	６６	１０．９ms	４３．７ms
５．10		２	６６	１０．９ms	４３．７ms
５．11	Prewittテンプレート型	３	１０７．２５	８７．４ms	３４９．５ms
５．12	オペレータ	２	７１．５	１３１．１ms	５２４．３ms

５．４　ＩＳＰの応用例

画像処理用ＬＳＩ－ＩＳＰを応用したシステムは、これ迄三つ公表されている。そのうちの二つは、画像の実時間処理に重点をおいて、ＰＥ増設方式を用いてシステムを構築している。他の一つは、システムの小型化に主眼をおいて、ＰＥ節約方式によりシステムを構築している。

ＰＥ増設方式を用いたシステムの一つに、汎用画像認識装置ＨＩＤＩＣ－ＩＰがある[16]。ＨＩＤＩＣ－ＩＰの外観とシステム構成を、それぞれ図５．１５と図５．１６に示す。図５．１６に示すように、ＨＩＤＩＣ－ＩＰは、システムプロセッサに１６ビットマイクロコンピュータ６８０００を、システムバスにＩＥＥＥ７９６マルチバスを用いている。ＨＩＤＩＣ－ＩＰ内の画像処理専用プロセッサＩＭＰ(Ｉmage Ｐrocessor)は、システムバスを介して与えられる６８０００の指令により動作する。画像処理用ＬＳＩ－ＩＳＰは、ＩＭＰの中のＶＬＳＩ内蔵プロセッサに３個搭載されており、３×３空間積和演算、８×１２パターンマッチングや、色彩距離による分類などの機能を実現している。（ＩＭＰ内の他の三つのプロセッサ――ヒストグラムプロセッサ、特徴抽出プロセッサ、ラベリングプロセッサ――には、ＩＳＰは搭載されていない。）表５．２にＩＭＰの処理機能を示す[11～13]。

ＰＥ増設方式によるもう一つのシステムは、超音波センサ併用高速画像処理装置である。これは、画像と超音波を併用して３次元物体を認識するシステムである。ＩＳＰを３個使用し、３×３のフィルタリングにより、輪郭強調と平滑化を高速に行っている[14]。

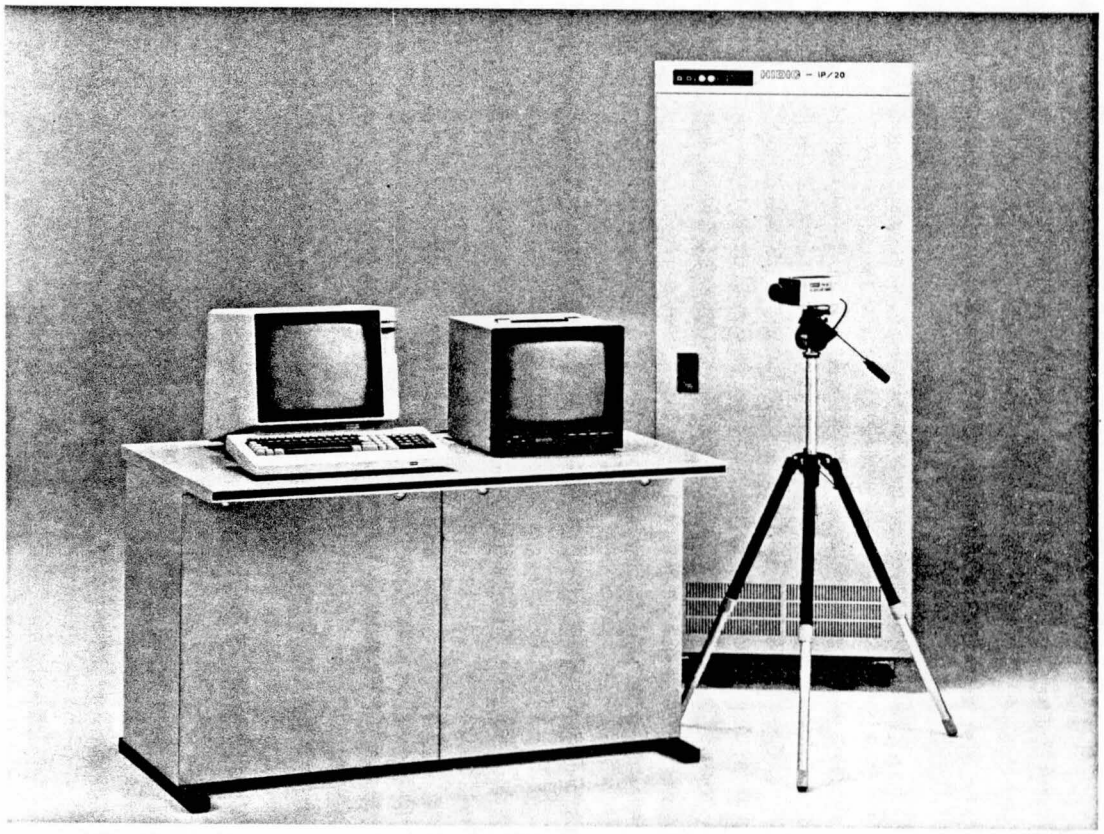


図５．１５　汎用画像認識装置ＨＩＤＩＣ－ＩＰの外観

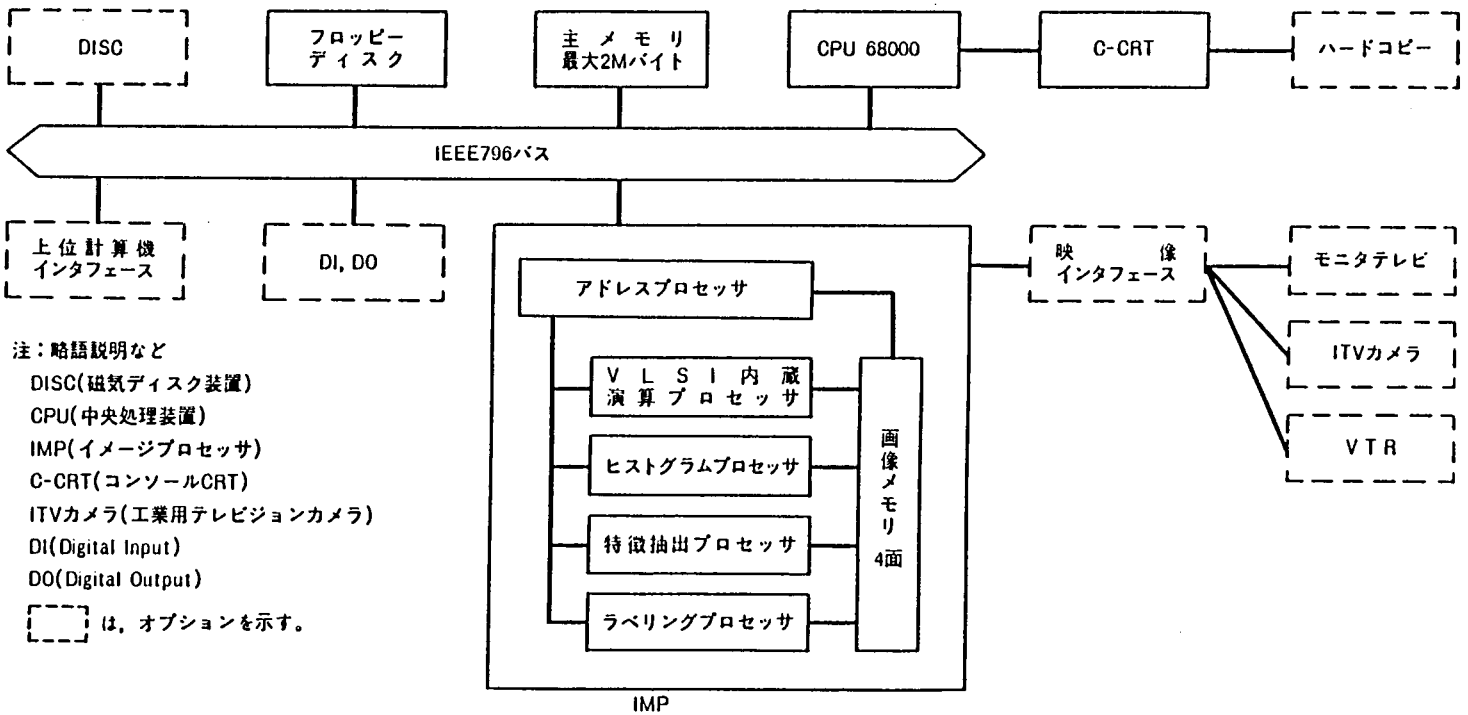


図５．１６　汎用画像認識装置ＨＩＤＩＣ－ＩＰのシステム構成

表５．２　画像処理専用プロセッサＩＭＰの処理機能

画像	処　理　内　容
２値	ラベリング，膨脹，収縮，輪郭抽出，細細化，点ノイズ除去，方向コード，端点・交点コード，パターンマッチング(８×12テンプレート)，ヒストグラム(面積，周囲長，物体存在領域抽出)，画像間演算
濃淡	濃度変換(γ補正，強調)，固定・浮動２値化 3×3空間積和演算，3×3非線形近傍演算，画像間演算，ヒストグラム(濃度分布，最大/最小濃度抽出)
色彩	色彩距離分類，単色変換
制御	ウインドウ，画面サイズ切換，マスク画像発生，カーソル発生

ＰＥ節約方式を用いたシステムの例としては、工業応用小型画像処理装置ＳＢＩＰ(Ｓingle Ｂoard Ｉmage Ｐrocessor)がある。ＳＢＩＰの外観を図５．１７に、システム構成と主な仕様を、それぞれ図５．１８と表５．３に示す。ＳＢＩＰではＩＳＰを４個用いて、３×３の濃淡フィルタリングや、最大３２×１６までの２値パターンマッチングなどの機能を実現している[15]。

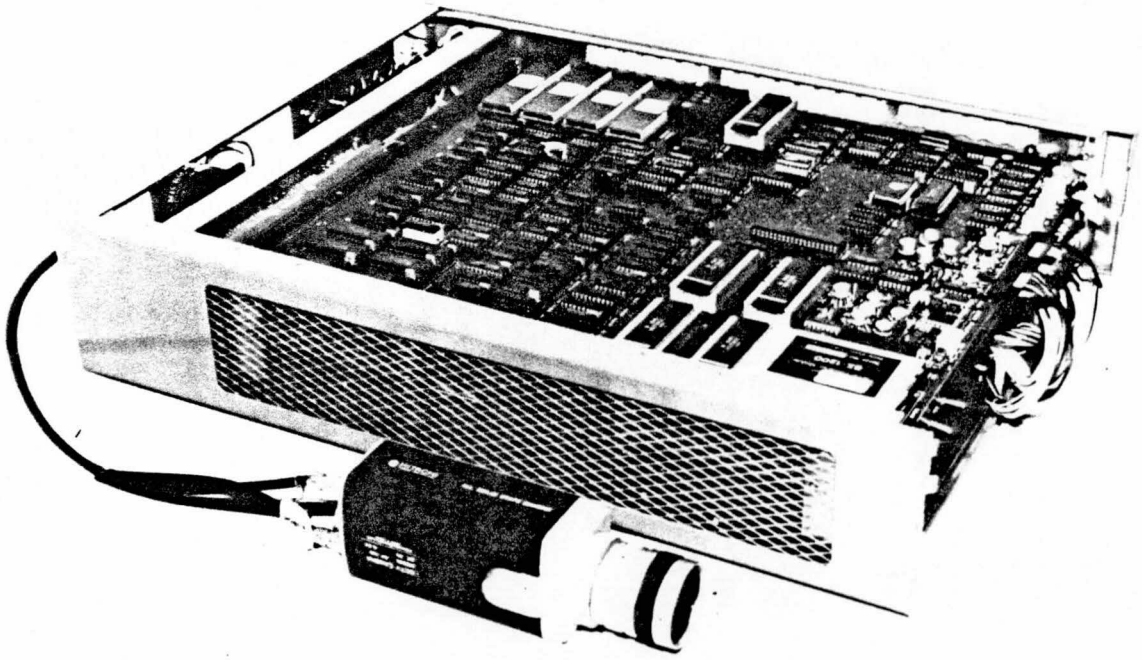


図５．１７　工業応用小型画像処理装置ＳＢＩＰの外観

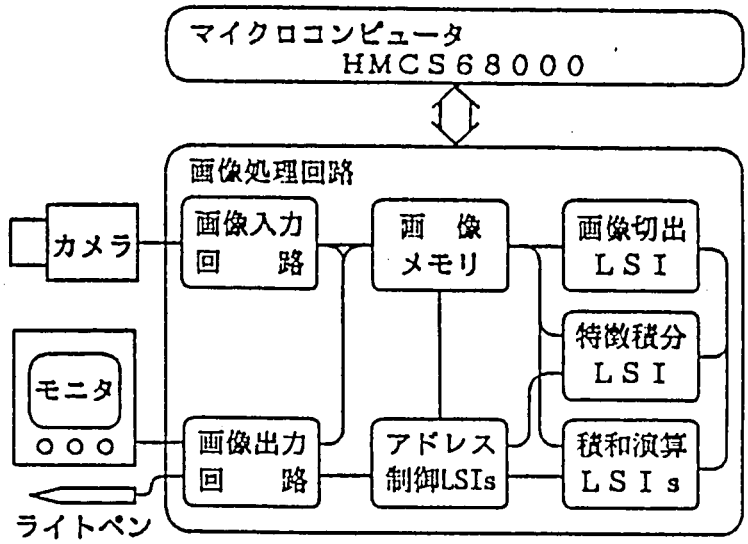


図５．１８　工業応用小型画像処理装置ＳＢＩＰのシステム構成

表５．３　工業応用小型画像処理装置ＳＢＩＰの主な仕様

項　　目	仕　　様
基本機能と性能	＊画像入力 ＊画像処理 ・２値パターンマッチング （８×８標準パターンで１１ミリ秒） ・濃淡パターンマッチング （４×４標準パターンで４４ミリ秒） ・濃淡フィルタリング （３×３係数行列で３３ミリ秒） ・特徴積分 （４４ミリ秒）、など ＊画像表示
プログラム言語	ＢＡＳＩＣ型画像処理言語
対話処理	・画像処理コマンドのダイレクト実行 ・ライトペンによるメニュー選択実行
画像メモリ	・２値画像メモリ(256×256)　１面 ・多値画像メモリ(256×256×８ビット)　１面
カメラ	最大 ２台 接続可能
Ｉ／Ｏポート	・シリアル(ＲＳ２３２Ｃ)　２ポート ・パラレル(１６ビット)　２ポート
外形寸法	幅４００×奥行５００×高さ１００ ｍｍ

５．５　まとめ

画像処理用ＬＳＩ－ＩＳＰを用いることにより、さまざまな画像処理機能をもつシステムを、コンパクトに構築することができる。ここでは、各種のカーネルサイズについての空間積和演算やパターンマッチング、１次微分オペレータ、さらにマルチマスクオペレーションとしてのPrewittテンプレート型オペレータ、などを例にあげて画像処理システムの構築例を示した。

ＰＥ増設方式によるシステムの構築例としては、カーネルが４×４、８×８の空間積和演算、およびカーネルが８×８、１６×１６のパターンマッチングをあげた。それぞれ、４個、１６個、２個、８個のＩＳＰを用いて構築できる。いずれのシステムも、２５６×２５６画素サイズの画像ならば、１０.９ｍｓでそれぞれの画像演算を実行でき、テレビ画像の実時間処理が可能である。

ＰＥ節約方式によるシステムの例としては、カーネルが４×４、８×８の空間積和演算、およびカーネルが３２×３２のパターンマッチングを例にあげた。それぞれ、１個、４個、８個のＩＳＰを用いることによりシステムを構築できる。いずれのシステムも時分割数は４であり、ＰＥ増設方式による場合と比較して４倍の処理時間を必要とする。つまり、２５６×２５６画素サイズの画像ならば、必要な処理時間は４３.７ｍｓである。この場合、テレビ画像の走査速度に追随して、実時間処理することはできない。

１次微分オペレータとしては、二つのオペレータを例にとってシステムの構築例を示した。それぞれ２個のＩＳＰを使用している。処理速度は、２５６×２５６画素サイズの画像で１０.９ｍｓであり、実時間処理も可能である。

マルチマスクオペレーションの例としては、８個の３×３マスクデータを用いるPrewittのテンプレート型オペレータを実行するシステムを示した。ＰＥ増設方式およびＰＥ節約方式により使用するＩＳＰ数は、それぞれ３個、２個である。処理速度は、２５６×２５６画素サイズの画像に対して、それぞれ８７.４ｍｓ、１３１.１ｍｓである。

以上のことから、たとえばＩＳＰを４個用いる場合、ＩＳＰの周辺回路をフレキシブルに作成することにより、ＰＥ増設方式で４×４空間積和演算と８×１６パターンマッチング、ＰＥ節約方式で８×８空間積和演算と３２×１６パターンマッチングなど、種々の画像演算を実行できる、柔軟性のある画像処理システムを構築することも可能である。具体的な例としては、ＩＳＰを３個用いて、３×３空間積和演算、８×１２パターンマッチング、色彩距離による分類、などを実現した汎用画像認識解析装置ＨＩＤＩＣ－ＩＰや、ＩＳＰを４個用いて、３×３空間積和演算、３２×１６パターンマッチング、などを実行できる工業用小型画像認識装置ＳＢＩＰ(Single Board Image Processor)などがある。前者はＰＥ増設方式を、後者はＰＥ節約方式を採用している。

５．６　第５章の参考文献

[1]Fukushima, T., Kobayashi, Y., Hirasawa, K., Bandoh, T., Ejiri, M. and Kuwahara, H. : An Image Signal Prosessor, IEEE ISSCC Digest of Technical Papers, Vol.26, pp.258～259 (1983).

[2]福島　忠、小林芳樹、平沢宏太郎、坂東忠秋、江尻正員：画像処理用ＬＳＩ－Image Signal Processor のアーキテクチャ、電子通信学会論文誌(C)、Vol. J66-C, No. 12, pp.959～966 (1983).

[3]福島　忠、小林芳樹、平沢宏太郎、坂東忠秋、柏岡誠治、加藤　猛：多機能性を実現する画像処理用ＬＳＩ－ＩＳＰのアーキテクチャ、情報処理学会論文誌、Vol. 25, No. ５, pp.728～735 (1984).

[4]福島　忠、小林芳樹、平沢宏太郎、坂東忠秋、柏岡誠治、加藤　猛：マルチマスクオペレーションを効率良く実行する画像処理用ＬＳＩ－ＩＳＰのアーキテクチャ、情報処理学会論文誌、Vol. 26, No. ２, pp.239～246 (1985).

[5]Fukushima, T., Kobayashi, Y., Hirasawa, K., Bandoh, T., Kashioka, S. and Katoh, T. : ISP A Dedicated LSI for Gray Image Local Operations, Proceedings of 7th International Conference on Pattern Recognition, Vol. 1, pp.581～584 (1984).

[6]Fukushima, T. : Image Signal Processor Computes Fast Enough for Gray-Scale Video, Electronic Design, Vol. 32, No. 20, pp.209～215 (1984).

[7]Ejiri, M., Uno, T., Yoda, H., Goto, T., and Takeyasu, T. : A Prototype Intelligent Robot That Assembles Objects from Drawings, IEEE Transactions, Vol. C-21, No. 2, p.161 (1972).

[8]Rosenfeld, A. and Thurston, M. : Visual Texture Analysis 2, University of Maryland Computer Science Center Technical Report 70－129 (1970).

[9]Prewitt, J. M. S. : Object Enhancement and Extraction, in Picture Processing and Psychopictorics, Academic Press, pp.75～149 (1970).

[10]福島　忠、加藤　猛：画像処理用ＬＳＩ－ＩＳＰとその応用、O plus E、No. ５, pp.76～86 (1984).

[11]小林芳樹、奥山良幸、福島　忠、平沢宏太郎、加藤　猛、久保　裕：ＬＳＩ化画像処理用装置の基本アーキテクチャ、情報処理学会第２６回全国大会講演論文集、pp.941～942 (1983).

[12]秦　清治、中川泰夫、浅野敏郎、福島　忠、加藤　猛：ファクトリーオートメーションにおける画像処理技術の応用、日立評論、Vol. 65, No. 12, pp.39～44 (1983).

[13]Fukushima, T., Kobayashi, Okuyama, Y., Katoh, T. and Kubo, Y. : High-Speed Image Processing System for Factory Automation, Proceedings of １９８４ International Conference on Industrial Electronics, Control and Instrumentation, Vol. 1, pp.444～447 (1984).

[14]武長　寛、金崎守男、坪井信義、鈴木優人、小林芳樹:超音波センサ併用高速画像処理の検討、電子通信学会技術研究報告ＰＲＬ８３－８４、pp.63～68 (1984).
[15]加藤寛次、柏岡誠治、植田博唯、江尻正員、福島　忠、大山祐一:工業用小型画像処理装置ＳＢＩＰ、日本ロボット学会学術講演会予講集、pp.263～264 (1984).
[16]小林芳樹、福島 忠、奥山良幸、加藤　猛、臼井敏雄：汎用画像認識解析装置ＨＩＤＩＣ－ＩＰシリーズ、映像情報、Vol. 16, No. 10, pp.19～24 (1984).

第６章　結論

ディジタル画像処理は、近年、電子計算機と半導体技術の進展を背景として、一般産業に幅広く適用されつつある。これまでは、画像の最小構成要素である画素(pixel)のおのおのに、１ビットの情報を与えた２値画像処理が主であったため、形状の識別や位置合せなど、比較的処理の容易な応用に限られていた。しかし、今後は、それぞれの画素に複数ビットの情報を与えた濃淡画像を用いて、目視検査の自動化やＸ線像の自動診断など、より高度な処理を必要とする分野へのニーズは高まりつつある。一方、一般産業へ適用してゆくためには、画像処理装置そのものが高性能であるとともに、小型でかつ安価でなければならない。そこで、高性能な画像処理用ＬＳＩ－ＩＳＰ（Ｉmage Ｓignal Ｐrocessor）を開発した。

画像処理用ＬＳＩ－ＩＳＰは、主に小型・安価の点から局所並列型とし、次の３点を開発目標とした。

（１）　空間積和演算に代表される局所画像処理を、ビデオレート（１６７ｎｓ／画素）で高速処理できる。

（２）　任意のおおきさのカーネルの処理が可能である。

（３）　ＬＳＩの制御信号を変更することにより、２値画像・濃淡画像の種々の基本演算を実行できる。

上記の方針に基づいて検討した結果、ＩＳＰのアーキテクチャを以下のように決定した。

（１）　ＬＳＩ製造技術上の制限から、並列動作するＰＥ（Ｐrocessor Ｅlement）の数は４個とした。

（２）　ビデオレートでの高速処理を達成するため、内部演算回路をパイプライン化する。

（３）　ＬＳＩを多数個を同時に用いる、もしくは、ＬＳＩ内のＰＥを時分割に使用することにより、任意サイズのカーネルの処理を実現する。なお、ＰＥを時分割処理する際には、カーネル切り出し用の遅延回路が不必要な画像操作方式－－スティック操作方式－－を新たに提案した。

（４）　種々の画像演算を実現するため、ＰＥへのデータ供給を柔軟な構成にした。さらに、それぞれのパイプラインステージの実行機能を独立にし、それらの機能の組合せにより、ＬＳＩ全体としての実行機能を実現する。これらの制御は、ＬＳＩ内蔵のプログラマブル制御レジスタの内容により決定される。

本論文では、さらに複数個のマスクデータを用いるマルチマスクオペレーションを効率よく実行できるよう、ＩＳＰのアーキテクチャを改善している。アーキテクチャの改善は、マルチマスクオペレーションを実行するとき、画像メモリの参照回数を極力少なくするよう、ＰＥへのデータ供給方式と、マスク演算間の処理方式について検討した。つまり、ＰＥへのデータ供給については、マスクデータの変更をＲＡＭの読み出しアドレスの変更により、また画像データの繰り返し供給を、ＳＲにフィードバック機構を設けることにより実現した。さらに、マスク演算結果間の処理においては、２値の比較回路のほかバイナリィカウンタを設けることにより、最大値もしくは最小値の摘出だけでなく、その値を生成したマスクデータの番号をも出力できる。

最後に、画像処理用ＬＳＩ－ＩＳＰを活用したシステムの構築法について、例を挙げて示した。ＩＳＰにより、高性能な汎用画像処理システムを、小型でかつ安価に構築することができる。

謝辞

本研究の推進に対しては、(株)日立製作所日立研究所、同中央研究所、同生産技術研究所、同大みか工場、同武蔵工場、日立エンジニアリング(株)、日立マイクロコンピュータ(株)など、多くの事業所や系列会社の多くの人々に、御指導および御助力頂いた。

特に、(株)日立製作所研究開発部長、高砂常義工学博士、同日立研究所長、川本幸雄工学博士と、(株)日立製作所大みか工場副工場長、桑原洋氏には、本研究の機会とその動機づけを、(株)日立製作所日立研究所第８部長、西原元久工学博士と、同第10部長、平沢宏太郎工学博士からは、本研究に対する大局的な御指導を、(株)日立製作所日立研究所主任研究員、坂東忠秋工学博士、および同研究員小林芳樹氏からは、日頃より具体的な御指導を頂いている。さらに、(株)日立製作所中央研究所主管研究員、江尻正員工学博士、同主任研究員、柏岡誠治氏、および(株)日立製作所生産技術研究所主任研究員、秦清治氏には、数々の有意義な御討論を頂いた。

また、京都大学工学部教授、長尾真工学博士には、本報告をまとめる際の御指導を頂いた。

以上の方々に対し、深甚の謝意を表する。